University of Alberta

SHAPE AND REFLECTANCE RECOVERY FROM INDOOR IMAGE SEQUENCES

by

Neil Aylon Charles Birkbeck

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2005

0-494-09128-2

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file *Votre référence*
ISBN:
Our file *Notre reference*
ISBN:

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Several successful methods for shape reconstruction from image sequences have been developed using a variational formulation. In this work we utilize the same framework, which leads to a Partial Differential Equation (PDE) describing the motion of an initial surface to a refined surface that is a better match to the input images. Motivated by the primary goals of reconstructing the object and the parameters to a reflectance model, we take advantage of known lighting conditions in the error measure. In particular, we assume that there is light variation, due to object rotation relative to the light source, allowing the recovery of shape in both textured and textureless regions. Additionally, we propose a method to filter out specular highlights, which allows the recovery of surfaces having non-Lambertian reflectance. Following recent work using an explicit surface parameterization, we apply the PDE refinement to a deformable mesh.

We test our method using images obtained from an easy to use capture setup. The setup is accessible to the average PC user, because the only hardware requirements are a camera, a light source, and a glossy white sphere. The capture setup provides images, camera calibration, light calibration, and silhouette images to be used in the refinement. The visual hull is used as a starting point for the PDE evolution. At the end of the refinement, our method outputs a triangulated mesh and the parameters of a Phong reflectance model represented in texture space.

Results on real and synthetic images demonstrate that this method is capable of recovering the geometry of textureless surfaces, and moderately textured surfaces, but is unstable in the recovery of deep concavities. Several examples on real sequences illustrate the applicability of our models in computer graphics applications, where the recovered objects are composed and rendered under novel lighting and view conditions.

Acknowledgements

I would like to thank all my friends and family for their endless support and encouragement during the writing of this thesis. Specifically, I would like to thank Keith Yerex, Dana Cobzaş, and my supervisor, Martin Jägersand, for valuable feedback and discussions along the way. In addition to the numerous discussions, I would like to thank Keith Yerex for helping me get my first summer research position. This research position sparked my interest in the field and introduced me to my current supervisor, who then convinced me to apply to graduate school. This work was funded by an Ivy A Thomson and William A Thomson Graduate Scholarship and a NSERC CGSM.

Table of Contents

1 Introduction 1

2 Previous Work 4
2.1 Shape From Silhouette 5
2.2 Depth/Disparity Map Representation 6
2.3 3D Mesh Representation 11
2.4 Voxel Based Representation 15
2.5 Level Set Representation 17
2.6 Discussion 19

3 Theoretical Fundamentals 21
3.1 Image Formation 21
3.1.1 Definitions 21
3.1.2 Pinhole Camera 22
3.1.3 Reflectance Models 23
3.1.4 Reflectance Fitting 26
3.2 Shape Refinement 27
3.2.1 Problem Definition 27
3.2.2 Shape Functional 28
3.2.3 Lambertian Object 31
3.2.4 Specular Object 33
3.3 Discussion 34

4 System 35
4.1 Capture Setup 36
4.2 Camera Calibration 37
4.2.1 Automatic Thresholding 38
4.3 Light Calibration 41
4.3.1 Detect Sphere Position 41
4.3.2 Compute Ray for Each Image of the Sphere 42
4.3.3 Intersect Rays 43
4.3.4 Compute Source Color 44
4.4 Shape from Silhouette 45
4.4.1 Silhouette Images 45
4.4.2 Marching Intersections Shape from Silhouette 46
4.5 Mesh Refinement 47
4.5.1 Deformable Mesh 47
4.5.2 Error Optimization 48
4.5.3 The Shape Optimization Algorithm 51
4.6 Texture Coordinate Generation 53
4.7 Reflectance Fitting 55
4.7.1 Interpolation Method 56

5 Experiments 58
5.1 Synthetic Objects 58
5.1.1 Varying Texture 59
5.1.2 Specular Objects 62
5.2 Real Objects 64
5.2.1 Reflectance Results 64

6 Conclusions 72

A Sampling the occluding contour of a sphere 74

Bibliography 76

List of Figures

2.1 Properties that can be used to classify the previous work, along with some representative classes. . . . 4
2.2 The silhouettes from multiple viewpoints restrict the volume of the recovered object. 5
2.3 A pair of stereo images (left, middle) and the corresponding disparity map (right) computed using a standard local window based stereo matching method. Bright values in the disparity map indicate features that are close to the camera. . . . 7
2.4 Three input images of a dog with different light directions and the recovered depth using standard photometric stereo. . . . 9
2.5 An example of voxel coloring, where the cameras are arranged in a circular pattern above the dog. The blue plane denotes the sweeping plane moving downwards through the volume. The consistency of each voxel is checked only once. . . . 15
2.6 Evolution of a level set using a shape from silhouette energy. . . . 18

3.1 The geometry of the pinhole projection model. . . . 22
3.2 Parameterization of a direction in the upper hemisphere at a surface point $\mathbf{p}$, with normal $\mathbf{n}$. . . . 24
3.3 An illustration of the unit length vectors used in the shading computation. . . . 26
3.4 An illustration of the curvature at a point on the surface S by examining the curves created by intersecting two planes with the surface. . . . 30
3.5 The cameras c_1 through c_4 all observe the same triangle on the current mesh. Neighboring cameras sample from overlapping regions, whereas samples from c_1 and c_4 do not overlap. . . . 32
3.6 The cameras c_1 through c_4 all observe the true surface, but the number of pixels occupied by the surface patch varies between the cameras. . . . 32

4.1 An overview of the system implementation showing the flow of information through several modules in the system. . . . 35
4.2 A snapshot of the capture setup. . . . 36
4.3 Our slightly modified version of the 3D S.O.M. pattern. . . . 37
4.4 The specular highlights of a white sphere rotating on the turntable are used to find the light position. . . . 41
4.5 The brightest pixel on the sphere in an image defines a ray, $\vec{r}(t) = \mathbf{o}_b + t\mathbf{d}_b$, that is intersected with the sphere to find the direction to the light source. . . . 43
4.6 The optimization seeks to minimize the squared distance of $\mathbf{x}_{light}$ to each of the 3D rays. . . . 43
4.7 An input image of a ping-pong ball (left) and the rendered sphere after fitting the sphere and light color (right). The blueish region on the left image denotes which pixels were filtered out as specular highlights. Because the ping-pong ball is modeled as a perfect sphere, the hemispherical inconsistency in the left image is not observed in the rendered image. . . . 44
4.8 A color $\mathbf{x}$ is part of the background if it lies within a certain distance of the ellipsoid defined by the background samples. The distance is computed by transforming the point into a space where the ellipsoid is a sphere. . . . 45
4.9 Computing the intersection point on the 3D ray, given the intersection point in an image. . . . 46
4.10 Sample points are chosen on an orthogonal lattice within each triangle. . . . 48
4.11 Computing the gradient at the circled vertex directly influences the adjacent triangles (marked with 1's), and influences the normals at the distance 2 neighbors (marked with 2's). . . . 49

4.12 An exaggeration of the displacement at a vertex $\mathbf{v}$, demonstrating how the neighboring vertex normals are affected in the gradient computation. This in turn affects the interpolated samples over the triangles at distance 2 from $\mathbf{v}$. 49
4.13 An example of the gradient computation by computing the consistency measure on the entire mesh. The neighboring triangles affect a single vertex, while distance 2 triangles partially influence the gradient on more than one vertex. 50
4.14 Several iterations of a silhouette preserving displacement applied to a smoothed object. 53
4.15 The binary mask of the piece (left) is to be inserted into the global occupancy mask (middle). The texture coordinate offset for the piece is obtained by finding the first available position for the piece, illustrated in the updated occupancy mask (right). . 54
4.16 A rendering of the texture coordinates (left) for an object (middle). The image on the right is obtained by texture mapping the model using the image on the left as the texture. 55

5.1 The names of the data sets and 2 of the 64 input images from each of the diffuse sequences. On the right is the texture used for the sequence. 59
5.2 The initial shape from silhouette mesh and the ground truth of the synthetic object. 60
5.3 The results of the synthetic diffuse experiments. For the first three surface types (DIFF, TEX0, and TEX1), the shape was successfully recovered. The high frequency texture in the TEX2 sequence caused some difficulties for the reconstruction, which is illustrated by the remaining volume in the concavities. 61
5.4 The names of the data sets and 2 of the 64 input images from each of the specular sequences. On the right is the texture used for the sequence. 62
5.5 Results for the specular sequences. Comparing the results in H50 to those of H50L (no specular filtering), we can see that the specular filtering helps produce a better reconstruction. The shape recovery was also successful in the H20 sequence, and the H50TEX1 sequence. 63
5.6 Two input images (top), synthetic images of the ground truth (2nd row), the initial volume (3rd row), and reconstruction results (bottom) from the Bunny sequence. The fine details start to become visible in the reconstructed results, mostly visible around the feet and between the legs of the bunny. 66
5.7 Two input images (top), the initial volume (2nd row), shaded reconstruction results (3rd row), and textured reconstruction results (bottom) from the TallHouse sequence. The refined results start to bring out the stairs in the base of the house, as well as the indentation by the chimney. The textured and lit renderings on the bottom row faithfully represent the true object. 67
5.8 Two input images (top), the initial volume (2nd row), the albedo computed for the initial volume (3rd row), shaded reconstruction results (4th row), and albedo of reconstruction results (bottom) from the ShortHouse sequence. Notice the blurriness of the texture in the 2nd row resulting from inaccuracies in the shape. After the refinement, the recovered texture is sharp (4th row), and novel views that are far from input views are possible (1st column of 4th row). 68
5.9 Some results for shape estimation and reflectance fitting of specular objects. From left to right: an input image, a shaded image, the diffuse texture, diffuse and specular texture at the same viewpoint, and a novel viewpoint with shadows and two lights. The renderings of each object are realistic, although the specular component is not as sharp as observed in the input images. 69
5.10 Multiple captures (real images of the two objects are given on the top row) were combined with some synthetic objects (the sphere and ground plane) and then rendered in Blender with multiple novel lights and a novel viewpoint (bottom). The vertices of the elephant have been modified to demonstrate the practical usefulness of the models. 70
5.11 The unique pieces of the real chess board (top) were captured individually. These pieces were then combined to produce a working chess game. The middle and bottom images show two lighting conditions of the chess game, where the objects are both casting and receiving shadows. 71

A.1 If $\mathbf{a}_1$ and $\mathbf{a}_2$ are on the occluding contour of the sphere then $\overline{\mathbf{a}_2\mathbf{o}} = \overline{\mathbf{a}_1\mathbf{o}} = r$, $\angle\mathbf{o}\mathbf{a}_2\mathbf{c} = \angle\mathbf{o}\mathbf{a}_1\mathbf{c}$, and $|\mathbf{c}\mathbf{o}| = |\mathbf{c}\mathbf{o}| \Rightarrow \triangle\mathbf{o}\mathbf{a}_2\mathbf{c} = \triangle\mathbf{o}\mathbf{a}_1\mathbf{c} \Rightarrow \overline{\mathbf{a}_2\mathbf{c}} = \overline{\mathbf{a}_1\mathbf{c}}$. 74
A.2 The occluding contour of a sphere is a circle on the sphere, which is projected into the image. 74

List of Symbols

$\mathbf{P}$	A camera projection matrix	5
Π	The projection operator	5
I	An image	9
$\mathbf{n}$	Surface normal	9
$\mathbf{K}$	The intrinsic parameters of a camera matrix	22
$\mathbf{R}$	The rotational component of the camera matrix	22
$\mathbf{t}$	The translational component of the camera matrix	22
f_x, f_y	The focal length of a camera	22
c_x, c_y	The principle point of a camera	22
α	The skew of a camera	22
k_1, k_2	Radial distortion coefficients	23
$\mathbf{k}_d$	Diffuse material color: $\mathbf{k}_d = (k_d^r, k_d^g, k_d^b)^T$	24
$\mathbf{k}_s$	Specular color of Phong model: $\mathbf{k}_s = (k_s^r, k_s^g, k_s^b)^T$	25
n	Specular hardness of Phong model	25
$\mathbf{a}$	The ambient light color: $\mathbf{a} = (a^r, a^g, a^b)^T$	25
$\mathbf{k}_a$	Ambient material color: $\mathbf{k}_a = (k_a^r, k_a^g, k_a^b)^T$	25
$\boldsymbol{\ell}$	Point (or directional) light source color: $\boldsymbol{\ell} = (\ell^r, \ell^g, \ell^b)^T$	26
$\mathbf{h}$	The halfway vector between the light and view direction	26
$n_{cameras}$	The number of images used in the refinement error measure	31
n_{spec}	The number of images assumed to contain highlights	33
$\mathbf{x}_{sphere}$	The position of the light calibration sphere	42
$\mathbf{r}_{sphere}$	The radius of the light calibration sphere	42
σ_{mesh}	The minimum distance between two edges in the deformable mesh	47
n_{planes}	The number of approximating planes used for texture coordinate generation	54

Chapter 1

Introduction

Three dimensional computer graphics models have been widely utilized in the entertainment industry, ranging from special effects in movies to the characters and worlds of immersive 3D games. More recently, 3D models are becoming an important part of heritage preservation and are making their way into educational museum content. As a consequence, the virtual heritage objects can easily be shared and distributed through the Internet. In many cases, the models are manually created by artists, but with the increasing use of 3D models, a large body of research has focused on more automated methods. Many of these methods rely on images to recover the 3D geometry and/or a model of the scene reflectance.

The automated methods can be roughly broken into two classes: the passive methods and the active methods. The passive methods typically use only image data for reconstruction. Within this class of methods falls many of the so called shape from X methods, including shape from shading, shape from silhouettes, shape from stereo, and shape from shadows. In particular, shape from silhouette uses silhouette information to reconstruct solid objects, whereas shape from shading uses surface shading. One common passive approach is shape from stereo, which uses the image position of a scene point in multiple images to triangulate the depth of the scene point. The position and orientation of the cameras is commonly required for the shape from stereo methods, but other work on the structure and motion problem has shown that the camera position/orientations can also be recovered directly from the images. This work on structure and motion offers the potential for low cost systems based on a single moving camera to automatically recover the scene structure and a color texture of the scene (such systems are demonstrated in [71] and [67]).

The other class of methods, the active methods, emit something into the world and use the sensed results to reconstruct the environment. For example, sonar sensors, which are commonly used for collision detection in robots, detect reflected sound waves. Other active approaches, more commonly used for 3D model reconstruction, include laser range finding and structured/unstructured light methods. Laser range finders reconstruct a dense depth map by scanning a beam of light over the scene and use triangulation to find the depth of scene points. The unstructured light methods typically project a textured pattern onto the scene, which can then be used with a stereo image

pair to triangulate the depth. This approach reduces the ambiguities in traditional stereo matching, especially in textureless regions.

In some applications the 3D models are rendered to produce new 2D images from arbitrary viewpoints, with the only requirement being that the new images are realistic. This inspired an alternative stream of research, which does not attempt to reconstruct a complete geometric representation of the scene, but rather is focused on reconstructing a representation suitable for rendering novel views. These methods are typically referred to as the image-based rendering (IBR) methods.

One example of IBR is the Lumigraph [32] (or the Light Field [53]), which treats many input images as samples of the light leaving the scene along a given ray. To render a new image, the ray associated with each pixel is used to index a representation of the input images, retrieving the light along that ray. Unlike the Lumigraph, view-dependent texture mapping uses a coarse geometry and multiple input images [20]. At render time, the texture for a given polygon is computed by blending the input images at viewpoints that are similar to the novel viewpoint. Using the nearest input images accounts for errors in the geometry, effectively simulating the parallax caused by the underlying surface geometry. Moreover, the blending of near images also reproduces specular highlights observed in the original input images. A more compact method for view-dependent texture mapping is accomplished by a principle component analysis on the input images, which are first warped to a consistent reference frame, as done with Eigen-Textures [65] and dynamic textures [14].

All of the above methods have the potential to ease the creation of 3D models, or in the case of IBR, to obtain a valid scene representation for rendering synthetic images. Unfortunately, for the automatic creation of general purpose 3D models, no individual method is ideal. Particular implementations of passive methods, e.g., shape from stereo, are prone to errors on textureless or glossy surfaces but are appealing because they are relatively inexpensive. Active methods, such as laser scanning, produce accurate dense structure, but the required hardware is fairly expensive and not accessible to the average user. A slightly cheaper alternative to laser scanning is the use of structured/unstructured light, but additional hardware is still required to project a pattern onto the scene. As IBR methods typically retain many images, memory consumption can become problematic. Furthermore, IBR methods usually have a limited geometric representation, implying the objects cannot cast or receive accurate shadows. On the other hand, some IBR approaches, such as panoramic images, are useful in many practical situations. One example is the use of panoramic images in real estate, where a home buyer can browse panoramas of available homes via the Internet. A downside, common to almost all the above methods, is that the reflectance or texture is recovered under current lighting conditions, implying that renderings are subject to similar light conditions.

In this work, our main focus is on the semi-automated capture of 3D models from images, where the acquisition system has the following properties:

1. The recovered models are applicable for use in typical computer graphics applications.
2. The shape recovery is accurate on a wide variety of surfaces.

3. The acquisition does not require expensive hardware.

By typical computer graphics applications we mean that the objects should produce renderings with realistic shadows, allow for rendering under novel lighting/viewpoints, and be easily composed with other synthetic and captured models. The second property is desirable so that a variety of objects may be reconstructed (e.g., uniform material, spatially varying surface properties, etc.). Finally, the above hardware requirements ensure the resulting system is also useful for the average PC user.

As a great deal of computer graphics models are used in the entertainment industry, we do not expect an acquisition method to completely replace manual modeling but instead to aid the manual creation. That is, we understand that these models may need to be animated or altered by artists to achieve a specific effect. Therefore, the recovered models need to be easily manipulated, ruling out the image-based approaches, which may be unintuitive or hard to manually modify.

Some of the active methods, such as laser scanning and active light, directly violate the hardware requirements. Other active methods, such as those requiring light variation (e.g., photometric stereo), and the passive methods are possible candidates. First, we review these candidate methods (Chapter 2). Then, based on the existing literature, we propose a method that attempts to satisfy the aforementioned requirements. This method is composed of the main contributions of this thesis:

- We present a Partial Differential Equation (PDE) based shape recovery method that utilizes known light variation to recover the shape of a variety of solid surfaces. The derivation and implementation are both based on previous work in the variational formulation of the image-based shape recovery problem. Our contribution comes in the form of an error function that utilizes known light variation and filters out specular highlights, allowing for the reconstruction of both textured and textureless surfaces. (Chapter 3).

- We have developed a capture system that recovers a 3D triangulated mesh and the parameters of a Phong reflectance model represented in texture maps. This system has been designed to be easy to use, and it requires commonly available hardware: a light source (e.g., table lamp), a camera, and a glossy white sphere (e.g., painted ping-pong ball) used to calibrate the light source. Therefore, the capture system is a practical way of providing the input to the shape recovery method, while satisfying the above mentioned properties (Chapter 4).

We then evaluate the effectiveness of our capture setup on a variety of synthetic and real image sequences (Chapter 5). These experiments demonstrate that the method is useful for recovering the shape of objects with varying surface properties. Results of relighting, shadowing, and composing several objects in a scene are also presented.

Chapter 2

Previous Work

Shape reconstruction from image(s) is one of the most active topics of research in computer vision; therefore, it is a topic for which many methods have been proposed. In this chapter we will give an introduction, and specific classification, to the existing methods and ideas in the literature. This review is only a sparse sampling of the previous work, but it gives a basic understanding of the types of image information used for shape reconstruction as well as the typical choices for scene representation during the reconstruction.

Classification Property	*Classes*
Scene Representation	depth/disparity map, 3D triangulated mesh, voxel grid, level set
Lighting Conditions	known, constant, accounted for, recovered
Information Used	shadows, specular highlights, shading, texture, silhouettes
Surface Material	textured, Lambertian, non-Lambertian
Number of Images Used	single, multiple

Figure 2.1: Properties that can be used to classify the previous work, along with some representative classes.

In order to present the previous literature in a meaningful manner, it is helpful to classify the methods based on a certain property. Some of the potential properties for classification are given in Figure 2.1. One natural property for classification is the type of representation used in the scene reconstruction. Common choices include disparity maps, 3D triangulated meshes, voxel grids, and level sets. Other distinguishing characteristics include the type of image information used, the class of surface materials the method can reconstruct, or other assumptions made by the methods, such as known lighting conditions. The type of image information is typically either surface texture, relied upon by some two camera stereo methods, or shading variation, utilized in shape from shading and photometric stereo. The class of surface materials reconstructed can be broadly divided into Lambertian/non-Lambertian or textured/non-textured.

All of the above are potential properties for classification, but for this work, we find the most natural classification to be the surface representation and the assumed knowledge on lighting conditions. We partition the light knowledge into 4 classes: *known*, the position and strength of the illumination is modeled; *constant*, the light is unknown, but assumed to be constant; *accounted for*,

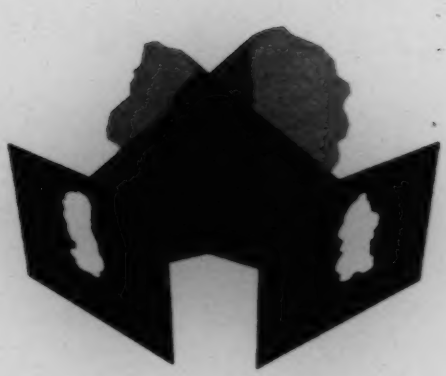

Figure 2.2: The silhouettes from multiple viewpoints restrict the volume of the recovered object.

the method accounts for light variation caused by object movement relative to the light source; and *recovered*, the parameters of an illumination model are recovered along with the shape. We also find it easier to discuss the shape from silhouette methods separately as they are occasionally used by other methods to obtain an initial estimate.

2.1 Shape From Silhouette

Shape from silhouette uses multiple silhouette images to recover a solid 3D object that produces the same silhouette as the object of interest in the input images (Fig. 2.2). The method takes as input a set of n binary silhouette images, S_i, where a $true$ (resp. $false$) denotes a pixel is within (resp. outside) the object, and the calibration parameters associated with the images. The separation of the object from the background is typically accomplished by using background subtraction or by using color segmentation on a solid color background. A 3D point $\mathbf{x}$ belongs to the object if and only if it projects inside the silhouette of each input image:

$$\bigwedge_i S_i(\Pi(\mathbf{P}_i\mathbf{x})) \tag{2.1}$$

where $\mathbf{P}_i$ is the camera calibration matrix for image i and Π is the projection operator (formally defined in Chapter 3).

The best possible reconstruction using only silhouettes is known as the *visual hull* [49]: the reconstructed volume from an arbitrary number of silhouette images. The visual hull is typically a good approximation of the true scene object, but features that are not observed in the silhouettes cannot be recovered. These features are typically indentations into the object. Although it is not practical to obtain the actual visual hull, using several images typically produces a volume that is close enough.

A simple method that uses voxels to reconstruct the visual hull was presented by Szeliski [86]. A typical implementation would be to initialize each voxel as belonging to the object and then

projecting them onto the input images to see if they belong to the object. A conservative carving would carve only those voxels that project to all background pixels in any image. Visibility does not need to be accounted for, and in the worst case, each voxel needs to be projected onto every input image to determine if it is carved or part of the object. For a particular set of input images, the accuracy of the reconstruction is limited to the voxel grid resolution.

Another method to reconstruct the approximate visual hull is to extrude the silhouette edges of each input image to get a 3D silhouette cone for each image. These silhouette cones can then be intersected in 3D to obtain the visual hull [4]. An alternative method for computing the polyhedral visual hull was proposed by Matusik et al. [61]. By projecting the 3D silhouette cone of an image onto the other input images, the polygon intersections are computed in 2D instead of being directly computed in 3D.

Alternative representations, such as the Marching Intersections (MI) data structure [75], have also been proposed for obtaining shape from silhouette [87]. The MI data structure contains three sets of rays, one set for each direction of the 3 principle coordinate axis. The rays in the x direction, are distributed on a lattice in the $y - z$ plane; the other two sets of rays are similarly defined. The rays are then projected onto the silhouette images, and the entry/exit points of the ray with the silhouette boundary are computed. The entry/exit points are quickly determined by scan-converting the projected line and keeping track of the pixel locations where $true$ and $false$ are crossed. These intersection points are then back-projected onto the world rays. Tarini et al. note that this method is capable of producing a more accurate shape than the voxel methods of similar resolution, while making more efficient use of memory [87].

2.2 Depth/Disparity Map Representation

We now move onto the methods that use a depth/disparity map as their surface representation. There is a great deal of literature pertaining to this representation as depth/disparity maps are commonly used in the binocular stereo methods. Most of the two-camera and multi-camera stereo methods fall into the class of unknown/constant lighting, and they typically rely on surface texture for successful reconstruction. The two-camera algorithms commonly deal with rectified stereo pairs, where the epipolar lines are aligned with the horizontal scan lines (see [72, 29] for details on the rectification). The task of these algorithms is to find the correct matching of pixels in the left image to the pixels in the right image (see Fig. 2.3). Once the matching is obtained, the reconstruction of the 3D point is given by triangulation [33]. As the match is constrained to lie on the same horizontal scan line, for each (u_l, v) in the left image, a match (u_r, v) is found in the right image. The change in the u direction, $\delta u = u_r - u_l$, is known as the disparity, and it is inversely proportional to the depth.

Scharstein and Szeliski present a review and taxonomy of recent two-camera stereo methods, suggesting that the algorithms can be broadly classified into local and global methods [78]. The local methods basically compare a window from the left image centered around (u_l, v) with a window in

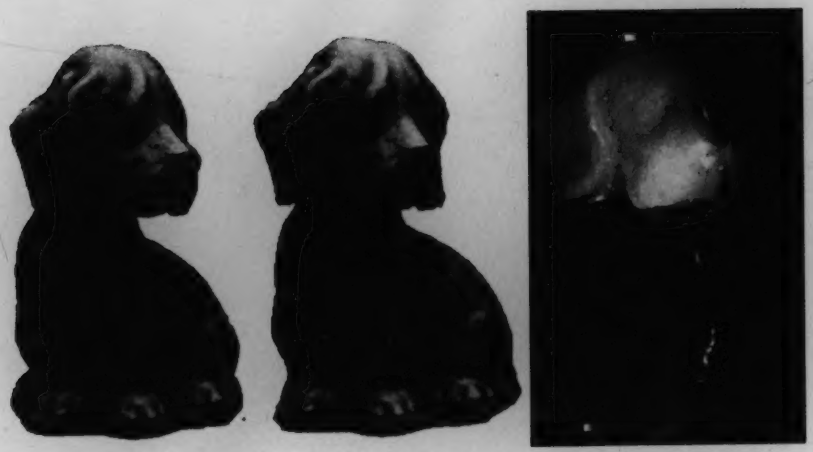

Figure 2.3: A pair of stereo images (left, middle) and the corresponding disparity map (right) computed using a standard local window based stereo matching method. Bright values in the disparity map indicate features that are close to the camera.

the right image centered around (u_r, v). A search along the scan line is performed to find the (u_r, v) whose window minimizes a similarity score between the window in the left image. This score is dependent on the method, with possible choices including the sum of squared differences (SSD) score, the sum of absolute differences (SAD) score, or a zero normalized cross-correlation (ZNCC) score. The methods are defined as local because each individual pixel in the left image chooses a corresponding pixel in the right image, independently of the neighboring pixels. There is some notion of smoothness, introduced by the choice of window size, but neighboring pixels are free to have widely differing disparities.

In contrast, the global methods formulate the two-camera stereo problem as a global energy function over the entire image that takes into account a per-pixel matching cost as well as a per-pixel smoothness cost [78]. The matching cost ensures that a pixel obtains a disparity, δu, that maps it to a pixel of similar color in the right image. The smoothness constraint ensures that neighboring pixels have similar disparities, which is typically true if the pixels observe the same object. The smoothness energy makes the global methods less sensitive to noise than the local methods, but it is an NP hard problem to recover the disparity map that gives the global minimum of the energy function [9].

Two camera stereo algorithms typically assume that the scene is Lambertian. With the addition of an extra camera, some of the problems caused by non-Lambertian highlights can be overcome. Bhat and Nayar propose a three camera setup, consisting of a left, right, and center camera [6]. Stereo matching then occurs between one of 3 possible pairs of images. If the quality of a match in a particular pair is poor, then that pair of images cannot be used, and one of the other two pairs of images is used instead. H. Zhang et al. also demonstrate that a trinocular method produces better

results than a binocular scheme [103]. Kolmogorov and Zabih have also extended their two camera global matching algorithm to take advantage of a multi-camera setting, where they again use graph cuts to optimize the error function [43].

Although most of the two-camera stereo techniques fall into the *constant* lighting category, some can be classified to lie within the *accounted for* category. Recall that by *accounted for*, we mean that the light has changed as a result of moving the object with respect to the light source. This light variation is common in turntable sequences where a single camera is used to capture multiple views of an object rotating on the turntable. The stereo methods that use ZNCC as their cost measure roughly fall into this category. ZNCC is a correlation measure for comparing two measurement vectors, and in the case of stereo, the measurements are two local windows. The ZNCC is computed by subtracting the mean from the measurement vectors, normalizing them to be unit length, and then taking the dot product between the two normalized vectors [22]. In this manner, the ZNCC can be seen as accounting for light variations [24]. Another example is the theoretically based cost function of Simakov et al. [82]. Their method properly accounts for the light variation on a Lambertian object due to object movement with respect to a fixed camera and unknown lighting conditions.

In the *known* light conditions category, we place the methods of photometric stereo and shape from shading. Although these methods typically recover a surface normal for each pixel, we place them in the *depth map* category, as the normal map can be integrated to obtain a depth map (see Frankot and Chellappa [27] for details of one such method for integration).

Unlike the above mentioned stereo vision, which requires two or more input images, shape from shading uses a single image of a surface with a uniform bi-directional reflectance distribution function (BRDF) as input. Assuming a directional light source, and ignoring inter-reflections and self-shadowing, the image of the object is dependent only on the orientation of the surface. The goal of shape from shading is then to recover the per-pixel orientation that produced the image. A reflectance map, R, which is a function of surface orientation, is commonly used to encode the surface material properties as well as the light conditions. In other words, R maps an orientation to an image intensity (or an RGB color). It is possible that two orientations map to the same intensity, so additional constraints, such as smoothness or integrability of the surface, are then used to compute the surface orientation at each pixel [106]. The reflectance map needs to be known in advance, so typically the object is assumed to be Lambertian; hence, the reflectance map is implied by the light direction. Other researchers investigate the use of non-Lambertian models in their methods for shape from shading. One such work is that of Ragheb and Hancock [74] whose algorithm is capable of iteratively removing the specular component, leaving a Lambertian image which shape from shading can then be successfully applied to.

Woodham introduced photometric stereo [94], a method that can recover the per-pixel orientation of a surface without any additional constraints. Instead of using only one image, Woodham suggests using several images of an object viewed from the same viewpoint but different lighting conditions.

Figure 2.4: Three input images of a dog with different light directions and the recovered depth using standard photometric stereo.

In the case of a Lambertian object, three lighting conditions can be used to recover the surface orientation at each pixel. Assuming $n \geq 3$ image measurements I_i, which were obtained under illumination from directional light sources, $\mathbf{l}_i \in \Re^3$, we have a set of equations of the form [94]:

$$\begin{bmatrix} \mathbf{l}_1^T \\ \mathbf{l}_2^T \\ \vdots \\ \mathbf{l}_n^T \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} \tag{2.2}$$

The normal $\mathbf{n} = (n_x, n_y, n_z)^T$ can be obtained if the light source directions span a 3 dimensional space. Actually, the solution vector is the normal times the diffuse albedo, which can easily be separated by normalizing the solution vector to be unit length. As the above system of equations is computed individually for each pixel location, photometric stereo can recover a varying diffuse albedo. In other words, Eq. 2.2 can be evaluated for each pixel, allowing the surface to be textured Lambertian (see Fig. 2.4).

As in shape from shading, photometric stereo can also handle non-Lambertian surfaces. For a uniform non-Lambertian surface, the orientation can be obtained if images of a reference object with a similar material are available [95]. If the surface normals for each pixel of the reference object are known (easy when the reference object is a sphere), then there is a set of intensity measurements for each normal. The orientation on the scene object is then obtained by looking up the surface normal with the same intensity measurements over the different set of illumination conditions. The idea of using reference objects has been more recently explored by Hertzmann and Seitz [35], who present a method that assumes that the BRDF of the scene object is a linear combination of the BRDF's of a set of reference objects.

Wolff and Angelopoulou present a method for recovering the shape of diffuse objects by using another type of photometric information [93]. This information comes in the form of two *photometric ratio* images. The photometric ratio images are obtained by using a stereo camera rig to capture two stereo pairs, one pair illuminated by each of two different point light sources. The left (resp. right) photometric ratio image is then obtained by simply dividing the intensity values of the left (resp. right) images. Although the lighting positions are unknown, we place this method in

the *known* light category, as the method requires differently illuminated images for the photometric ratio. The authors demonstrate that the photometric ratio of a surface point should be the same in the stereo pair, so a traditional disparity search is then used to obtain the disparity map by matching the values in the photometric ratio images. The matching of ratios is shown to improve the stereo matching across textureless regions, but it seems that self-shadowing could reduce the effectiveness of the method.

Another interesting approach for shape recovery using multiple images taken under different lighting conditions is presented by Magda et al. [57]. The method uses Helmholtz reciprocity to overcome problems typical stereo matching algorithms have with view-dependent highlights. For any BRDF, Helmholtz reciprocity states that interchanging the incoming and outgoing directions should produce the same value. The authors discuss a multi-image system, and they implement a two image system: one image is taken with a camera and light source (not located at the camera), and another image is taken after switching the positions of the light and camera. Using Helmholtz reciprocity, the view-dependent effects are stationary on the object in the images. Therefore, traditional stereo matching can then be performed on their input images, with the results not being affected by specular highlights. In later work, the multi-image system was implemented via a camera and light source attached at opposite ends of a rotating disk [108]. The depth is then recovered from a single camera location, which is not restricted to the positions of the input images. The addition of the extra image pairs allows this method to recover surface normals as well as depth information. Whenever the matching fails, for example, in areas with little surface texture, the recovered normals can be used to obtain an accurate shape.

Most of the discussion so far has concentrated on recovering surface geometry using either stereo or shading/photometric variation, but there have also been approaches for merging the two. One attempt to merge shape from shading and binocular stereo was proposed by Cryer et al. [16]. The authors compute shape from shading and binocular stereo individually on the input images, and these individual results are then merged in frequency space. The merging uses the low frequency component of the stereo results and the high frequency component of the shading results, implying that errors in the stereo will propagate to the end result. In some sense, the two methods are not cooperating, in that shading cannot make up for errors in the stereo results, or vice versa. An alternative method, proposed by Lange, offers a more cooperative approach for merging photometric stereo with binocular stereo [48]. Additionally, common simplifying assumptions on reflectance/illumination are lifted from photometric stereo, and a Phong model of reflectance and point light sources are used instead.

Falling in the category of *recovered* illumination conditions is the work of Brooks and Horn, which estimates the light source direction as well as the surface orientation in shape from shading [10]. Another example is the work of Hayakawa, who proposes a method to recover the light source direction and surface normals in photometric stereo using a rank constraint to factor the observa-

tion matrix [34]. Unfortunately, this reconstruction is only known up to an arbitrary 3 degree of freedom transformation of the light sources and the normals. The transformation is known as a generalized bas relief (GBR) transformation, which is discussed in detail by Belhumer et al. [5]. Another method for surface and light source reconstruction using photometric stereo is presented by Georghiades [31]. In his work, a simplified Torrance-Sparrow reflectance model is used, enabling the reconstruction of a non-Lambertian surface and the light sources up to a binary ambiguity.

Another work that recovers light sources while using a depth map as the surface model is the work of L. Zhang et al. [104]. Their method is initialized by tracking a few feature points over an input sequence. These tracked points give the camera parameters for an orthographic camera, the feature points in 3D, and the surface normals at the feature points. These points and surface normals are used to initialize the estimate of light conditions, which is assumed to be a directional light and ambient term per image frame. The dense per-pixel depth and normal computation is then constrained by the recovered light and camera parameters. The authors note that the dense 3D points are inaccurate in textureless regions but the normal information is accurate. So instead of using the 3D surface points, the dense normals are integrated to obtain a depth map, and the resulting depth map is aligned with the 3D coordinates of the sparse feature points. This method is unlike the typical photometric stereo methods presented above, where the camera is stationary and the lighting is changed. In this case, the object is rotated in front of a stationary camera and light, and the intensity variation on a surface point in 3D is used to obtain the photometric information and hence the surface normal. As a general capture method, this can be seen as more convenient than photometric stereo because only one physical light source is needed.

2.3 3D Mesh Representation

Triangulated surface meshes are a common representation for 3D objects in computer graphics, and several researchers have proposed methods which use this representation for shape reconstruction. Fua and Leclerc note that this representation allows for merging of multiple types of information, e.g., shading/texture, as well as being an easy representation to take visibility into account [28]. These methods typically take some input mesh as a starting point and refine this mesh according to some error measure, which produces a mesh that is consistent with the images. Aside from the choice of error metric, some of the other issues involved when using a mesh based representation include the following: Should the error be measured on the mesh, or in the images? Should the triangles in the mesh be regular sizes, or permitted to have varying sizes/shapes? For the first question, there is no straight answer. Even though recovering a mesh that best matches the input images is the goal of the reconstruction, it may be easier to implement a method that measures the error on the surface. For the second issue, it may seem ideal to have a mesh with irregular sizes, allowing a high resolution reconstruction only when the underlying surface is detailed, but it may be more complicated to implement than the alternative method. In this section, we see that the existing mesh

based methods explore the above options, as well as many other possible directions.

We begin our review in the category of *constant* lighting, which does not necessarily follow the chronological order of appearance of the mesh based methods. As the mesh based methods are trying to find a model that matches the input images, an error measure denoting the similarity of the model to the scene must be chosen. A common error measure for Lambertian scenes is a variance based color consistency measure. For a point on the surface of the object, the colors from all cameras in which the point is visible are collected; the error metric is the variance of these colors. Zhang and Seitz use this error measure on sample points within each triangle of the mesh, and they assume that the initial estimate encloses the true scene object [105]. The error at each of these surface points is used to compute a displacement for the surface point. If the error is above some threshold, then the current point is assumed to be far from the true surface, and the displaced point lies in the direction of its negative normal, essentially shrinking towards the true object (given the assumption that the initial object encloses the true scene). Otherwise, when the error is below the threshold, a gradient based method is used to move the point towards the true surface. Given the new displaced points, the vertices of the triangles are then moved to best approximate this displacement. Triangles that cannot fit the displacement of its sample points are then subdivided so as to better fit the sample points. Mesh regularity is ensured using Garland and Heckbert's mesh simplification algorithm, which also reduces the resolution of the mesh in regions with little geometric detail.

A similar approach that is developed theoretically using Partial Differential Equations (PDE's) is presented by Duan et al. [21]. The error measure is essentially a variance based consistency measure, which is computed on a tangent plane at the mesh vertices, instead of using sample points on the triangles. The iterative update, obtained as the solution to the PDE, moves the vertices in the direction of their normal with a speed that is dependent on the current error, the gradient of the error, and the curvature of the surface. The speed also includes a constant term that effectively forces the mesh inwards when the error function is non-zero and the current surface has little gradient/curvature information. This force is similar to the constant shrinking displacement used by Zhang and Seitz [105]. The search is initialized with a low resolution mesh that encloses the object, which is subdivided as needed during the optimization. Unlike many other mesh based approaches, their mesh implementation also handles topological changes, allowing the recovery of objects having different genus than the initial shape.

Isidoro and Sclaroff present a texture space method for recovering a triangular mesh [37], where the texture space is a 2D parameterization of the mesh. For a given mesh, the best texture image is obtained by warping the input images to texture space and taking a weighted average. The warping uses the 3D geometry to map visible triangles in the input images to the corresponding triangle in texture space. If a triangle is visible in a particular view, then the weighting for that triangle in the view is based on the projected area of the triangle in that image. Otherwise, the weighting for that view is zero. The error in texture space is easily computed by warping the input images to texture

space, subtracting the current texture image, and taking a weighted average. The refinement is accomplished by sampling the points on the surface which have large error and performing an epipolar search to find a more consistent surface point. Their estimate is initialized using the visual hull, so the search for a more consistent surface point is limited by the visual hull. Instead of directly moving these randomly selected surface points to their more consistent location, the recovered offsets are used to determine a free-form deformation that can be applied to the entire mesh. In addition to the contribution of the refined points, the resulting free-form deformation is also influenced by a silhouette preserving contribution. That is, points which are believed to belong to the visual hull are used to restrict movement in the areas of the current model which are potentially correct. Again the Garland-Heckbert algorithm is used to refine the mesh. The above process is iteratively applied to obtain the final reconstruction.

Vogiatzis et al. take a Bayesian approach to formulating the mesh reconstruction problem, and then they use simulated annealing to optimize the mesh [90]. Unlike the mesh methods presented so far, the error function is computed as a sum of the error in every pixel in every image. To compute the error in image space of image i, with calibration $\mathbf{P}_i$, all input images are warped to the image plane of I_i using $\mathbf{P}_i$ and projective texture mapping the current mesh. The average of these warped images, taking visibility into account, is an image W_i. The reconstruction seeks to make these synthetic images, W_i, best agree with the corresponding input images, I_i. Mesh regularity is imposed in the prior, which tries to keep the surface locally planar, while preferring a mesh with a smaller number of vertices. In order to optimize with simulated annealing, the authors describe the mutations used in their work: some mutations move a vertex in a random direction, some move a vertex in the direction of the gradient, and some move a vertex along further distances using a correlation based search. Other mutations are targeted at the smoothness (prior) and consist of swapping/removing edges or introducing new edges.

An alternative method proposed by Vogiatzis et al. attempts to reconstruct the depth along the surface normal from an initial low resolution mesh [91]. The depth is sampled on the triangles of the mesh, and the problem is formulated similar to the global stereo methods. For each sample on a triangle, there is an energy that attempts to find the best depth, and there is also a smoothness term suggesting that the neighboring samples should be at similar depths. The smoothness constraints are extended across triangles to ensure a consistent looking mesh. The depths are optimized by Loopy Belief Propagation, in a multi-resolution subdivision of possible depths. The authors note that the advantage of such a representation over standard disparity maps are the ease at which it handles many images and solid objects.

Although most of the above works are essentially iterative methods used to refine the surface, others have proposed more direct methods. One such example is the work of Esteban and Schmitt [22], who propose a ZNCC carving approach. As discussed in the *disparity/depth map* category, this method is classified to belong to the *accounted for* light category because it uses the ZNCC.

Given an initial starting mesh obtained from shape from silhouette, the method first detects vertices on the mesh that are not on the true surface. This is done using a robust correlation score on the tangent plane to the object at the vertex. Each vertex that fails the correlation test is assigned a carving direction: the direction is aligned with the line through the vertex and the camera center of the median camera that the vertex is visible in. A correlation method that uses the nearest cameras and filters out poor camera pairs is used to find a distance to carve in the carving direction, with the search being limited by the initial shape. The authors note that the method cannot find a carving depth in regions where the surface is lacking texture.

We are unaware of other mesh-based methods that take into account light changes that are a result of a moving the object with respect to the lighting. The error metric of Simakov et al. [82], may directly extend to a mesh based implementation.

Known as one of the original mesh-refinement papers [105], is the work of Fua and Leclerc [28], which falls into the *known* illumination classification. The authors use a triangulated mesh to combine stereo and shading information. They assume that the light direction is known in advance, and that this lighting is fixed over the input images (i.e., the camera is moving with respect to the light and the object). They propose an error function that is a linear combination of a stereo term, a shading term, and a smoothness term. The stereo term is essentially the variance measure mentioned earlier, evaluated at equally spaced sample points on each triangle. Given the known light conditions and a current mesh estimate, the albedo at a surface point can be obtained. The albedo is then used in the shading component of the error function, which tries to ensure that neighboring triangles have similar albedos. The smoothness term essentially promotes local planarity of the surface. For each triangle, the shading term is weighted based on how well the triangle is approximated by a single albedo. On the other hand, the stereo term is inversely weighted. This ensures that the stereo term is active when texture is available; otherwise, the shading term is dominant. The optimization is performed using a conjugate gradient based method, which, as pointed out by the authors, requires an accurate starting position. For this reason, the mesh is initialized with standard stereo data.

Yu et al. present a method for recovering the shape of a uniform non-Lambertian object being illuminated by an arbitrary number of known directional sources [99]. Taking advantage of the mesh based representation and the known lighting, they properly model occlusions as well as shadows. They assume that the object to be reconstructed is star-shaped and parameterize their mesh using spherical coordinates. Movement is constrained to be along the radial component. Using an initial input mesh (i.e., shape from silhouette), they interleave the fitting of a Phong model of reflectance with the mesh refinement. After the reflectance model has been fit, the refinement is accomplished by optimizing the parameters of the mesh to minimize the difference between the model and the images. A multi-resolution and extrapolation scheme is used to speed up the optimization.

Although none of the discussed methods directly recover the light conditions, a mesh obtained from one of the above methods captured under a stationary object and moving camera may be usable

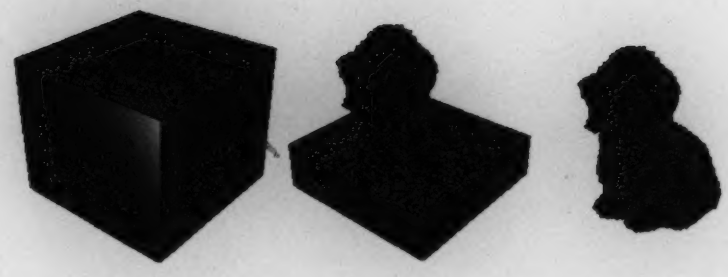

Figure 2.5: An example of voxel coloring, where the cameras are arranged in a circular pattern above the dog. The blue plane denotes the sweeping plane moving downwards through the volume. The consistency of each voxel is checked only once.

in other algorithms that estimate lighting given shape (e.g., the work of Nishino et al. [66]). One method that is somewhere between the *recovered* lighting category and the *accounted for* lighting category is the method of Yu et al. [100]. Their method recovers the shape along with a combined model of reflectance and illumination. The combined model is called a View Independent Reflectance Map (VIRM), and as the name suggests, it is related to the concept of a reflectance map used in shape from shading. The VIRM is a representation of a uniform surface material (which may be non-Lambertian) lit by a particular illumination. The view-dependent effects of the material are encoded in the VIRM, but only for the illumination at the time of capture. Furthermore, the VIRM can be visualized on other geometries, i.e., the material of the scene object is viewed on the other geometry subject to the illumination in the original capture. Unfortunately, the lighting/material properties in the VIRM cannot be separated. The recovery of shape using the VIRM is similar to the authors other work, mentioned above [99], in that the VIRM estimation and shape refinement are interleaved.

2.4 Voxel Based Representation

As mentioned in the shape from silhouette review (Section 2.1), voxel based representations can be used as a straightforward method to obtain the visual hull. Voxels are simply carved away from an initial solid volume to reveal the visual hull. Voxel based representations have also been used for reconstructions that rely on cues other than silhouettes. For example, Seitz and Dyer introduced *voxel coloring* [80], which is a method that attempts to find a set of voxels that could have produced the colors observed in the images. Ideally, under the Lambertian assumption, with a stationary object relative to the light source(s), a point on the surface of the object should project to the same color in all input images that the point is visible. In the presence of sensor noise and calibration errors, the point should project to a set of colors that are similar, taking into account any potential sources of error. If instead of a point we consider a voxel, a similar reasoning holds, although instead of projecting to an individual pixel, the voxel will project to a set of pixels. When these pixel sets

agree with a consistency measure, the voxel is thought to be part of the surface, and the voxel is said to be consistent. One possible consistency check is to threshold the variance of the observed pixel colors. The final surface should be composed of a set of voxels that are consistent. The process of voxel coloring is then to carve away the inconsistent voxels until only the consistent voxels remain. The remaining volume is referred to as the *photo hull*, and it is the maximal set of voxels that is consistent with the given images [45].

To avoid checking the consistency of a voxel whenever visibility changes (i.e., a consistent voxel becomes visible in another view when an inconsistent voxel is carved), Seitz and Dyer develop an algorithm to work on a subset of potential camera configurations. The constraint on cameras is that no surface point can lie within the convex hull of the camera centers [80]. This simple constraint allows the algorithm to check the consistency of each voxel only once, by visiting the voxels in a particular order (see Fig. 2.5 for an example). The restriction on camera placement has since been removed [44, 17].

Instead of simply using a consistency check on the pixels that observe the voxel, Zabulis and Daniilidis propose the use of a correlation score [101]. For a pair of cameras, they optimize a correlation score on the tangent plane of each voxel by searching the space of possible surface normals. Changing the surface normal affects the tangent plane, and the true surface normal should produce the tangent plane having the best correlation score. A small set of k cameras having the best pairwise correlation scores is found and used to refine the surface normal estimate. Unlike the previously mentioned algorithms, the surface is not carved away nor is visibility explicitly accounted for. Instead, after a voxel's score/normal have been computed, the surface is extracted by a 3D operator that is based on the Canny edge detector.

Some approaches have been proposed to handle non-Lambertian surfaces in voxel coloring [13, 97, 8]. Chhabra [13] and Yang et al. [97] propose similar methods that allow a structured variation on the observed colors. The argument is similar to what follows. When the camera is moving and the lighting is constant, the colors observed on a surface point can be represented as the sum of two components: a constant diffuse color and a view-dependent specular component. Assuming a white specular color, the observations should be on a line in RGB space, where the line goes from the diffuse color to the color of the light. If the observed colors lie near this line, then the voxel is part of the surface. This approach will work with some shiny surfaces, but will not work with mirror like surfaces. For such surfaces, an alternative specular approach of Bonfort and Sturm is more appropriate [8].

Other researchers propose to use the voxel space as a location to merge stereo data [24, 60, 23]. Matsumoto et al. use a voxel based approach to obtain the visual hull as an initial estimate, and then they perform a stereo matching to refine the shape [60]. For each input image, a *partial surface* facing the image is extracted by searching along a ray from the camera center through the voxel volume to find a voxel on the ray that minimizes a matching score. This voxel is then assumed to

belong to the surface. The score of each voxel is computed using a SSD consistency score in the neighboring k images. They do not explicitly handle occlusions, but instead they assume that a point is visible in some set of k adjacent images. After the *partial surfaces* have been computed, a vote is cast for the voxels behind the partial surface. The final voxel volume is extracted by thresholding the votes of the voxels in the volume.

In a similar manner, the methods of Esteban and Schmitt also use a voting scheme [24, 23]. In their case, for an input image I_i they use the neighboring k images and a robust ZNCC score to find the potential depth of the surface along a ray through every pixel in I_i. If the score at the potential depth is above a correlation threshold, then a vote is cast for the voxel containing the point. A triangulated mesh is then used to extract the surface from the voxel volume. The triangulated mesh, a 3D snake, moves along a linear combination of the gradient of a smoothing energy, the gradient of a silhouette energy, and the gradient of the voxel volume.

Other voxel based approaches require *known* lighting information [92, 89]. Weber et al. [92] use a turntable and capture images of a scene object lit by three different light sources, whose positions are calibrated in advance. Instead of using the standard variance consistency measure used to carve voxels, their error metric incorporates the light information and is valid for Lambertian scenes only. Given the light source positions, the error metric is taken as the residual after fitting a normal and diffuse albedo to the observed image intensities (somewhat similar to the photometric stereo method presented in Eq. 2.2). During the carving process, the known light positions allow for shadow information to also be computed. Therefore, only visible and lit image points are used in the error metric.

Another voxel method utilizing illumination variation is the work of Treuille, Hertzmann, and Seitz [89]. Their work extends the *example based photometric stereo* approach of Hertzmann and Seitz [35] (briefly reviewed in Section 2.2) to a voxel based representation. Again, the method requires images of calibration spheres having similar material properties as the scene. In the case of a single material, the measurements observed on a voxel are used to look up an orientation with the most similar observed colors. If the colors corresponding to the looked up orientation do not fit the observed colors well, then the voxel is carved. The multiple material method is similar, except that a linear combination of the different material colors at a particular orientation is fit to the observed colors.

2.5 Level Set Representation

We now move onto shape reconstruction algorithms that use level sets as their representation. We will focus our discussion on the methods developed for multi-camera reconstructions of solid objects, therefore we will not consider the use of level sets for disparity/depth reconstructions.

In the level set formulation, instead of some explicit parameterization of the surface, the surface is represented as the level set of an implicit function (e.g., $\Phi(x, y, z) = 0$). For a solid object, Φ also

Figure 2.6: Evolution of a level set using a shape from silhouette energy.

characterizes the points belonging to the object as those points for which $\Phi < 0$ and those outside the object as $\Phi > 0$. Level set methods are the numerical methods to solve surface evolution (described by PDE's) using the level set representation. The existence of such numerical methods and the ability of level sets to change topology during their evolution are two appealing characteristics of this representation.

Faugeras and Keriven were the first to use level set methods for multi-camera solid object reconstructions [25]. They took a variational approach and formulated the shape from multiple images problem as the reconstruction of a surface that minimizes a stereo matching cost, integrated over the surface. Following fundamental principles in variational calculus, the functional gives rise to a Partial Differential Equation (PDE) that describes the motion of a surface as a function of time. The desired surface, the one that minimizes the functional, can be obtained by evolving an initial surface through the motion described by the PDE (see Fig. 2.6). At this point, an explicit representation, such as a triangular mesh, could be used (an example of this approach was given in Section 2.3, namely [21]), but instead the authors used the level set methods [69].

Faugeras and Keriven originally used an accumulation of pairwise cross-correlation scores, measured on the tangent plane of the object, as their matching cost. Other researchers have proposed different error measures and use PDE's similar to those obtained by Faugeras and Keriven. Slabaugh, Schafer, and Hans use a simpler variance based consistency measure, which is computed over image regions that observe a particular voxel [83]. Jin et al. propose another error measure to overcome specular reflection [41]; their measure takes the median score over all possible two image ZNCC scores computed on a surface patch.

In later work, Jin et al. propose another variational formulation of the multi-camera stereo problem [40]. A new matching cost that is valid for non-Lambertian scenes is also presented. The previous matching costs projected a surface patch on the tangent plane of the object into multiple views, and then used a variant of cross-correlation of these image measurements for the matching cost. Instead of performing a pairwise correlation, the new measure first flattens the image measurements into the columns of a matrix, denoted the *radiance tensor*. Their error measure is based on the rank of this matrix. This cost function should also account for lighting changes due to object motion, so we classify this method as belonging to the *accounted for* class.

Some work with the level set representation involves assumptions of known lighting. For example, Jin et al. propose a variational method that takes advantage of a single known light position in order to reconstruct solid objects [39]. They propose partitioning the object into two regions: a textured region and a textureless region. Their method then operates on textureless regions, while the correlation based method of Faugeras and Keriven can be used on textured regions. As the lighting is assumed to be fixed and the camera moving, the color of a point (textured or not) in two images should appear the same. The textureless regions are then assumed to have uniform albedo, making the reconstruction similar to shape from shading, with the primary difference being the use of multiple images of the object.

In addition to recovering the shape of textureless scenes, later work by Jin et al. also recovers a model of the illumination [38]. This work is an extension of their previous variational work in *stereoscopic segmentation* [98] and *stereoscopic shading* [39]. Stereoscopic segmentation attempts to segment foreground objects from the background using multiple images, essentially recovering the approximate visual hull in the process. The extended work assumes that the light can be modeled as a collection of point sources, an ambient term, and a uniform hemispherical term. The evolution of the surface and the estimation of the illumination parameters are then interleaved to obtain the final reconstruction. As the albedo of the object cannot be separated from the light source strength, the albedo is assumed to be white. In addition to automatically segmenting the object from the background using multiple images, the use of shading information should allow for concavities to be recovered.

2.6 Discussion

With each of the surface representations comes advantages and disadvantages. In the case of shape from shading and photometric stereo, the depth/disparity map representation is the most natural representation. For binocular stereo, the depth/disparity map is also an appropriate choice as there is little to gain by a voxel or mesh based representation.

In the case of multi-image reconstructions, the depth/disparity map becomes less appealing. A single depth map could be used, but then the reconstruction may become biased to the chosen camera. Alternatively, if a depth map is reconstructed for each camera, then merging multiple representations and ensuring consistency among the data can become complicated. In either approach, visibility constraints can also become problematic.

The problems with visibility are easily overcome by choosing one of the alternative representations. Successful methods for multi-image reconstruction using each of these world centered representations have been discussed, but mesh and level set representations have some advantages over a voxel representation. One advantage is that smoothing or regularizing terms can easily be integrated into a mesh or level set. Another advantage is that the mesh and level set representations give readily computable surface normals. As the surface normal is directly tied to the surface shading, the

normal can be exploited when the lighting is known. Nevertheless, methods have been developed to take advantage of known lighting information while using a voxel representation [92, 89].

Ignoring the representation used, some observations can be made regarding the use of light information in the shape reconstruction process. Firstly, the methods that assume unknown lighting, or do not directly require light knowledge, are the most applicable to outdoor and uncontrolled environments. These methods typically perform best when the scene is sufficiently textured. Secondly, using light variation allows for the shape reconstruction in textureless regions. Unfortunately, the methods that introduce light variation are typically limited to laboratory settings, where light variation is obtained by moving a calibrated light source, by using several physical calibrated sources, or by moving the object and fixing the position of the source. Thirdly, some knowledge of the illumination conditions (or at least light variation) is necessary to recover reflectance parameters of the surface. Even in the recovery of the surface albedo, some knowledge of the light is necessary to undo the shading.

These are the observations that influenced the design of our reconstruction method. Specifically, we have chosen a variational approach, which is implemented using the mesh representation. Furthermore, we require known light variation to aid the reconstruction of textureless surfaces and to fit the reflectance parameters. A major influence of the design of the capture system presented in Chapter 4 was to provide this light variation, while keeping the capture setup as simple as possible.

Chapter 3

Theoretical Fundamentals

In this chapter we review the essential background pertaining to image formation and reflectance modeling, followed by a description of the central problem of this work: the shape reconstruction problem. After our formal definition of the general shape reconstruction problem, we use existing variational methods to formulate a solution. The general reconstruction solution is then specialized to work with textured Lambertian objects, followed by a method for dealing with specular objects. The particular contribution of this work is to utilize known light conditions in the reconstruction process, which allows the recovery of surfaces with both textured and textureless regions.

3.1 Image Formation

In this section, the necessary background of the image formation process as well as some definitions of terminology used throughout this work will be described.

3.1.1 Definitions

As this work is rooted in computer vision, a set of images will be the primary input. Input image i will be denoted as I_i, and the RGB triplet at row v and column u in image I_i will be denoted by $I_i(u, v)$. The red, green, or blue channel of an image is denoted by I_i^r, I_i^g, I_i^b, so that $I_i^r(u, v)$ means the red component of location $(u, v)^T$ in image i. Similarly, in the context of material properties, coefficients for a particular color channel will be denoted with a superscript, e.g., c^r, c^g, c^b or the more general c^λ. Bilinear interpolation is assumed for color retrieval at location $(u, v)^T$, for which u or v is not a natural number.

The pinhole model of a perspective camera will now be detailed. This is followed by an improvement of the model to compensate for radial distortion. The camera model gives a geometrical meaning of what is observed at a given pixel location. We then present a brief review of surface reflectance models, which gives a basic understanding of why a pixel at coordinates $(u, v)^T$ is a particular color.

3.1.2 Pinhole Camera

In the pinhole camera model, light from the portion of the scene in front of the camera passes through an infinitely small *focal point* and intersects the image plane, giving a 2D image. As illustrated in Fig. 3.1, the projection of a 3D scene point is found by intersecting the line through the scene point and the focal point with the image plane, which in this case is the plane $z = -f$. The size of an object in the image is proportional to the focal length of the camera, f, and inversely proportional to the distance of the object from the focal point along the optic axis. By convention, the image plane is assumed to be in alignment with the x-y axis of the cameras coordinate system. Therefore, the optic axis is aligned with the z-axis, and the depth of a point is given by its z coordinate.

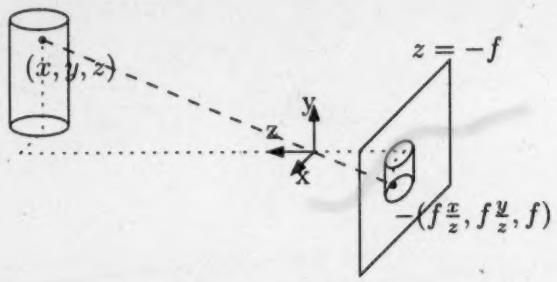

Figure 3.1: The geometry of the pinhole projection model.

The general pinhole camera allows for an arbitrary placement of the camera in a world coordinate system and an affine parameterization of the image plane. The position and orientation of the camera are given by a 3D Euclidean transformation consisting of a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 translation $\mathbf{t}$. Using homogeneous coordinates, the perspective projection of a 3D point $\mathbf{m} = (x, y, z, 1)^T$ is given by [33]:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{m} = \mathbf{P}\mathbf{m} \qquad (3.1)$$

The 3×4 matrix $\mathbf{P}$ is the projection matrix associated with an image taken by a camera. If the matrix $\mathbf{P}$ is known for a particular image, the image is said to be calibrated. The matrix $\mathbf{K}$ holds the intrinsic parameters of the camera including the focal length, f_x, f_y; the skew, α; and the principle point, $(c_x, c_y)^T$. A non-zero skew, α, suggests non-rectangular pixels, but the pixels are typically rectangular so we will restrict its value to be zero [33]. The final image coordinates, $(u, v)^T$, are obtained by dividing the coordinates of the point by the homogeneous component:

$$\Pi(\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}) = \begin{pmatrix} x'/z' \\ y'/z' \\ 1 \end{pmatrix} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \qquad (3.2)$$

Radial Distortion

In real cameras, light does not pass through a single infinitely small focal point. Instead, there is a circular opening with finite diameter letting light through, and a lens system is used to focus the light on the image plane. The lens is not directly modeled by the pinhole camera, so real cameras tend to deviate from the pinhole model. One common form of deviation is radial distortion: distortion as a function of the radial distance from the center of the lens [33]. The radial distortion can be expressed as a function of normalized image coordinates, where the normalized image coordinates $(x_n, y_n, 1)^T$ for a point $\mathbf{m} = (x, y, z, 1)^T$ are:

$$\begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = \Pi([\ \mathbf{R} \quad \mathbf{t}\ ]\mathbf{m}) \tag{3.3}$$

Following Zhang [107], radial distortion is approximated using two terms:

$$\begin{aligned} x'_n &= x_n(1 + k_1 r^2 + k_2 r^4) \\ y'_n &= y_n(1 + k_1 r^2 + k_2 r^4) \end{aligned} \tag{3.4}$$

Where $r = \sqrt{x_n^2 + y_n^2}$ is the radial distance, and k_1,k_2 are the radial distortion coefficients. The point $(x'_n, y'_n, 1)^T$ then undergoes the transformation by the intrinsic parameters $\mathbf{K}$ to obtain the image coordinates.

If image I was taken with a camera having radial distortion coefficients k_1 and k_2, then the image I' with no radial distortion and the same intrinsic and extrinsic parameters as I can be obtained by a simple rectification.

3.1.3 Reflectance Models

The perspective camera model gives geometric information about which pixel a scene point will be observed in, or conversely which ray in world space corresponds to a pixel. The other form of information available at an image pixel is its color, which is determined by the interaction of light with the surface point observed by the pixel. Reflectance models offer a way of understanding the interaction of surface and light. Some of the reflectance models used in the literature will now be discussed.

The appearance of an object is determined by the interaction of the surface of the object with the current illumination conditions as well as the relative position of the viewer with respect to the object. This interaction of light on the surface is commonly modeled with the five dimensional bi-directional reflectance distribution function (BRDF):

$$f^{\lambda}(\theta_i, \phi_i, \theta_o, \phi_o) \tag{3.5}$$

The BRDF is a local property of a surface, defined on the upper hemisphere about the surface normal of each point on the object (see Fig. 3.2 for an illustration of the parameterization of directions at a point). Given a wavelength of light, λ, the BRDF returns the fraction of light reflected from a

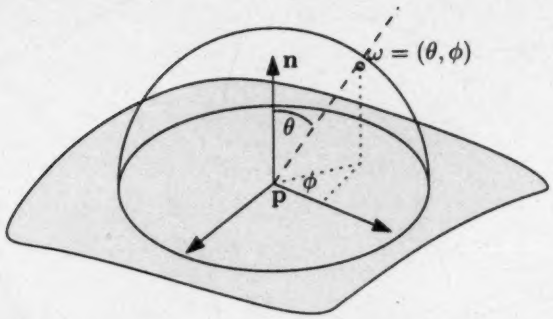

Figure 3.2: Parameterization of a direction in the upper hemisphere at a surface point p, with normal n.

particular incoming direction, $\omega_i = (\theta_i, \phi_i)$, in the outgoing direction, $\omega_o = (\theta_o, \phi_o)$. Instead of representing the BRDF as a function of wavelength, the 5 dimensional BRDF is often approximated by three separate 4D functions, one for each of the color channels: f^r, f^g, and f^b.

If there is only one light source, modeled as either a point light source or directional source, the color observed on a point is given by [1]:

$$E^\lambda(\theta_o, \phi_o) = f^\lambda(\theta_i, \phi_i, \theta_o, \phi_o)\ell^\lambda(\theta_i, \phi_i)\cos(\theta_i) \tag{3.6}$$

where $\omega_i = (\theta_i, \phi_i)^T$ is the direction from the surface point to the light source, and $\ell^\lambda(\theta_i, \phi_i)$ is the amount of light arriving at the surface. Note that this formulation allows for the light source to be either a directional source or a point light source. In the case of a directional source, the amount of light reaching any lit surface point is not dependent on the distance from the source. On the other hand, for a point source, $\ell^\lambda(\theta_i, \phi_i)$ encodes the degrading energy arriving at the surface, which is a function of the distance from the surface patch to the source. This fall-off can be modeled as being inversely proportional to the squared distance from the surface patch to the source [26]. The contribution of a finite number of light sources (which are visible from the surface) to the outgoing radiance at ω_o can be computed as a sum of the individual contributions. In the limit, all light in the environment (reflected and direct light) can be taken into account by integrating Eq. 3.6 over the upper hemisphere of the point, but this integration is typically too complicated for many Computer vision and Computer graphics applications.

By considering a single light source and ignoring the indirect light, we have simplified the shading model. Another dimension for simplification lies within the BRDF itself. Many researchers have developed simpler parametric functions that describe the BRDF [47, 55, 7, 70, 88]. The simplest of these models is to assume that the BRDF is constant: $\mathbf{k}_d=(k_d^r, k_d^g, k_d^b)^T$. This results in the Lambertian model, where light arriving at a surface from a particular direction is reflected equally in all directions. The Lambertian model effectively models matte objects, such as clay, where the shading observed is a result of the foreshortened contribution of the light source (i.e., as a result of $\cos(\theta_i)$). Substituting the Lambertian BRDF into Eq. 3.6, the color observed on the surface is then:

$$E^\lambda = k_d^\lambda \cos(\theta_i)\ell^\lambda(\theta_i, \phi_i) \tag{3.7}$$

Although the Lambertian model successfully models matte materials, shiny objects exhibit a view-dependent reflectance that strays from Eq. 3.7. For instance, shiny surfaces tend to reflect more light in the perfect reflection of the incoming light about the surface normal. With such surfaces, when the view direction (i.e., the output direction) is close to this perfect reflection direction, the surface has a highlight. This specular reflectance is typically modeled as an additive component to the Lambertian model. Such specular models include the Phong [70], Blinn-Phong [7], Torrance-Sparrow [88], and the Lafortune model [47]. For instance, combining Phong's view-dependent component with the Lambertian model in Eq. 3.7, and plugging the BRDF into Eq. 3.6 gives the following:

$$E^\lambda = (k_d^\lambda + \underbrace{\frac{k_s^\lambda \cos(\theta_r)^n}{\cos(\theta_i)}}_{Phong}) \cos(\theta_i)\ell^\lambda(\theta_i, \phi_i) \tag{3.8}$$

where θ_r is the angle between the reflected incoming direction and the outgoing direction, $\mathbf{k}_s = (k_s^r, k_s^g, k_s^b)^T$ is the specular color, and n is the specular exponent. A larger value of n signifies a sharper specularity, whereas a smaller value of n gives rise to a specularity observed on a surface point through a wider range of views.

Lewis points out that the Phong BRDF does not obey the natural properties of a true surface [55]. More specifically, Lewis notes that the Phong model is not energy conserving[1] nor does it obey Helmholtz reciprocity[2]. Nevertheless, the Phong model has been commonly used in Computer graphics because of its simplicity.

Up until now, the simplified shading models we have discussed ignored the contribution of the indirect light. In its simplest form, the contribution of indirect light can be modeled as another additive component. The conglomerate Lambertian, specular, and ambient shading model becomes [26]:

$$E^\lambda = (k_d^\lambda \cos(\theta_i) + k_s^\lambda \cos(\theta_r)^n)\ell^\lambda(\theta_i, \phi_i) + k_a^\lambda a^\lambda \tag{3.9}$$

where a^λ is a constant denoting the amount of ambient light of color λ present, and $\mathbf{k}_a = (k_a^r, k_a^g, k_a^b)^T$ is an ambient coefficient of the surface material.

Instead of modeling the reflectance in the local coordinate frame of a surface point, Eq. 3.9 is typically modeled using unit length vectors in a common coordinate system (Fig. 3.3). Letting $\mathbf{n}$ be the surface normal at a point, $\mathbf{l}$ be the direction to the light source, and $\mathbf{v}$ be the view direction (analogous to the outgoing direction), Eq. 3.9 becomes:

$$E^\lambda = (k_d^\lambda(\mathbf{n} \cdot \mathbf{l}) + k_s^\lambda(\mathbf{v} \cdot \mathbf{r})^n)\ell^\lambda + k_a^\lambda a^\lambda \tag{3.10}$$

[1] Energy conserving means that the total outgoing light does not exceed the total incoming light
[2] Helmholtz reciprocity implies $f^\lambda(\omega_i, \omega_o) = f^\lambda(\omega_o, \omega_i)$ for all ω_o, ω_i

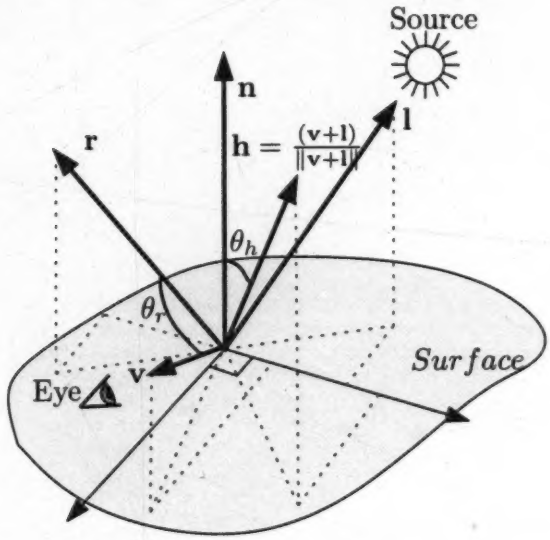

Figure 3.3: An illustration of the unit length vectors used in the shading computation.

where $\mathbf{r} = 2\mathbf{n}(\mathbf{n} \cdot \mathbf{l}) - \mathbf{l}$ is the perfect reflection of the incoming light vector about the normal. We have also dropped the angular index of ℓ^λ, as here we consider only one source of interest.

Replacing $k_s^\lambda(\mathbf{v} \cdot \mathbf{r})^n$ in Eq. 3.10 with $k_s^\lambda(\mathbf{n} \cdot \mathbf{h})^n$, where $\mathbf{h}$ is the half-angle:

$$\mathbf{h} = \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|}$$

the Phong model becomes the Blinn-Phong model. In this work, we use Equation 3.10 with the Blinn-Phong substitution as our shading model.

Throughout this work, the information from a digital image will be used as if it were a direct measurement of the scene irradiance. This is not true for real cameras, where the final image intensity is a non-linear function of the scene irradiance [63]. There exists methods to recover and undo this mapping, some based on multiple images of the scene, where the aperture/exposure [63, 19] or illumination [81] is varied, or by utilizing a single image of a known color calibration pattern [12]. Therefore, for accurate application of the ideas in this work, the actual scene irradiance rather than raw image intensities should be used.

3.1.4 Reflectance Fitting

Given a set of input images, an object surface, and illumination information, it is possible to recover the parameters of a reflectance model for the surface. Of course the method of fitting a reflectance function is dependent on both the chosen reflectance model and the assumptions about the lighting environment. Previous work in this area covers the various possibilities of reflectance models, such as the Phong model [77, 2, 76], the Torrance-Sparrow model [30, 36], or for estimating the BRDF

itself [62]. While most of the existing work covers the estimation of reflectance models under simple lighting, methods have also been developed for more complicated situations [96].

In our case, for low parameter BRDF models, given enough observations, the parameters can be estimated efficiently using an indirect iterated linear approach [36] or by a more direct non-linear method [47]. For example, the indirect methods essentially try to classify the observations into ones where only the diffuse component is expected to be observed. The diffuse color can then be fit to these samples. The specular contribution is then fit to the remaining points by first subtracting the diffuse contribution. This classification/estimation process is then iterated. On the other hand, the non-linear approaches seek to directly minimize the squared error between the observed samples and those predicted by the reflectance model.

One of the take home messages of the existing literature is that a dense estimation of the specular parameters is not always possible. In this regard, existing methods for parametric BRDF estimation differ in how the specular parameters are computed for the surface points in which the observations do not allow a direct fitting. One approach is to use an interpolation scheme to interpolate the specular parameters of the surface points whose parameter estimation is possible [77, 76]. Another approach, proposed by Lensch et al. [51], clusters the surface of the object into materials, and then fits a BRDF model to each of the clusters. This allows for a more accurate estimate of BRDF parameters, while a true spatially varying surface is obtained by fitting a linear combination of the cluster models to each surface point.

3.2 Shape Refinement

We now consider the shape refinement problem. We start with a general formulation, which is then specialized to work with Lambertian objects under known light variation. This is followed by a filtering method to aid the recovery of specular surfaces.

3.2.1 Problem Definition

The shape recovery problem takes as input a set of n images, $I = \{I_i | i \in 1 \dots n\}$; the associated calibration, $\mathbf{P}_i$; the illumination information, L_i; and outputs a shape, S, and corresponding reflectance parameters that best agree with the input images.

In the most general case it may be that nothing is known about the illumination conditions. In our specific work, we assume the illumination conditions consist of a single point light source and an ambient term. The object is assumed to be moving relative to the source, so the images contain illumination variation. Further, we assume that the surface reflectance parameters are implied by the surface. That is, given some desired parametric reflectance model, the parameters of the model can be fit using the surface, the surface normal (implied by the surface), and the imaging conditions given as input. Therefore, the shape recovery problem is to recover a shape, such that the shape and its implied reflectance parameters best agree with the input images.

3.2.2 Shape Functional

In this work, we use the same variational formulation of the shape recovery problem as Faugeras and Keriven [25]. That is, we wish to recover the surface, S, that minimizes the following functional:

$$F(S) = \int\int_S g(\mathbf{x}, \mathbf{n}) dS = \int_v \int_u g(\mathbf{x}, \mathbf{n}) \|\mathbf{x}_u \times \mathbf{x}_v\| du dv \tag{3.11}$$

Where $\mathbf{x} = (x(u,v), y(u,v), z(u,v))^T$ is a point on the surface and $\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}$ is the surface normal at point $\mathbf{x}$. The function g is a photo-consistency function that measures the consistency of a surface point with the images that observe the point. In this work, we are interested in investigating a photo-consistency function of the following form:

$$g(\mathbf{x}, \mathbf{n}) = \sum_i h(\mathbf{x}, \mathbf{n}, \mathbf{P}_i, L_i) \|I_i(\Pi(\mathbf{P}_i \mathbf{x})) - R(\mathbf{x}, \mathbf{n}, L_i)\|^2 \tag{3.12}$$

Where R is basically a graphics rendering equation, returning the color of the point $\mathbf{x}$ under lighting conditions L_i. The function h is a weighting function, typically taken to be the visibility of point $\mathbf{x}$ in image i. In fact, the function R is a function of the entire surface, as it should take into account any inter-reflections, and more importantly the visibility of the point $\mathbf{x}$ to the light source. A similar argument regarding the function h requiring the entire surface can be made. We will ignore these subtleties for the remainder of this discussion.

The function R also encodes the reflectance model at a point $\mathbf{x}$ on the surface, and may be any BRDF model, but for practical issues we are interested in parametric reflectance models. Recall we are assuming that the parameters of the reflectance model are implied by the surface shape, the input images and the available illumination information. Therefore, given that Eq. 3.11 is only dependent on the surface shape, the parameters of this BRDF model are obtained by a fitting process that in fact seeks to minimize Eq. 3.12.

The photo-consistency model in Eq. 3.12 does not necessarily generalize constraints on the observations at a point, such as the one proposed by Jin et al. [40], as it explicitly requires a BRDF model to be fit to the colors. A more general form of Eq. 3.12 for shape optimization would be similar to the one suggested by Kutulakos and Seitz [45]. Their measure does not assume a particular model of reflectance; instead, given the viewing geometry and light conditions, their approach simply measures the possibility of observing a set of colors on a point. A measure of this form would also generalize the constraint used by Jin et al. as well as the measure given in Eq. 3.12. On the other hand, as we are also interested in obtaining a parametric model of reflectance, we believe that the measure in Eq. 3.12 is appropriate. Furthermore, any other assumptions on the scene properties, such as the scene consisting of a uniform material, can be embedded in the function R.

A similar consistency function to Eq. 3.12 was used in the *stereoscopic shading* work of Jin et al. [39], but they assume a constant albedo over a region of the surface. Although our error function is essentially the same as theirs, Jin et al. assume the light is fixed with respect to the object during

the capture. Under these circumstances, the authors assume that the albedo is constant over a region of the surface, and that all color variations observed on this region are due to shading. This region is then refined using a shading error function, and another error measure is used on the textured regions. Another similar error function was used in the mesh based work of Yu et al. [99], but again they use a moving camera and assume that the object has a uniform non-Lambertian surface material. Instead, in our work, we assume that the object is moving relative to the light source (as done by Weber et al. [92]), and therefore, we allow for a true spatially varying albedo.

During the capture, motion of the object relative to the light source is easily accomplished using a turntable capture setup and a stationary light. In our system, we utilize two full rotations, each having a different light position to ensure sufficient light variation (see Chapter 4 for details). Following observations in other work [104, 92], we feel that this variation is sufficient to allow for the use of a spatially varying albedo during the capture, without affecting the reconstructed geometry. That is, even for a uniform Lambertian object, a spatially varying albedo will not be able to explain the shading effects in all the images, implying that the shading must come from the surface itself. This is much like the advantage that photometric stereo has over shape from shading, except that like stereo, we have to establish correspondence across multiple images.

Finding the surface, S, can now be done in a similar manner as done by Faugeras and Keriven. First, the PDE corresponding to the Euler-Lagrange equation of Eq. 3.11 must be derived, and then the minimization can be accomplished using level set methods. Alternatively, an explicit mesh representation can be used to perform the minimization, as done by Duan et al. [21]. In fact, Faugeras and Keriven derive the PDE for the general form of Eq. 3.11, which can be used directly with our consistency measure.

The PDE derived by Faugeras and Keriven contains higher order terms resulting from the general form of g being a function of $\mathbf{n}$. Instead of using the full PDE, complete with the higher order terms, we use a simplified PDE, which is accurate for a g that is only a function of surface position $\mathbf{x}$. This is essentially the same PDE described by Caselles et al. [11] and is similar to those used by Duan et al. [21]:

$$\frac{\partial S}{\partial t} = (2g\kappa c_{reg} - \nabla g \cdot \mathbf{n})\mathbf{n} \tag{3.13}$$

where κ is the mean curvature. The first component of the motion in Eq. 3.13, $2g\kappa c_{reg}$, is essentially a smoothing term, reducing the mean curvature of the object, whereas the second component ensures the evolution decreases the error function on the surface. The c_{reg} coefficient was introduced to weight the contribution of the smoothing term.

The mean curvature, κ, at a point on the surface is the average of the principle curvatures at the point. For a given point on the surface, consider a plane going through that point and containing the surface normal at that point (two such planes are illustrated in Fig. 3.4). The plane intersects the surface creating a curve that lies on both the surface and the plane. For any such plane, the normal curvature is defined as the curvature of the plane curve at the particular surface point. The principle

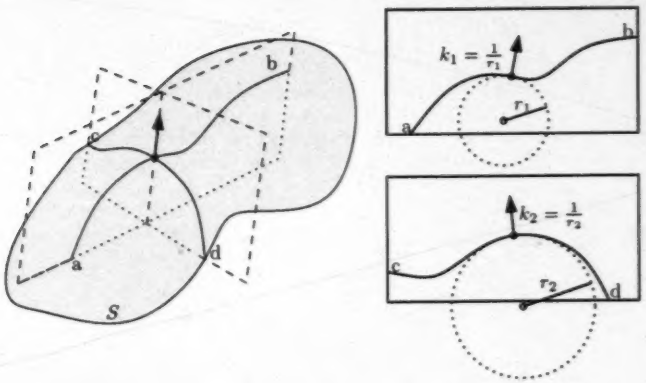

Figure 3.4: An illustration of the curvature at a point on the surface S by examining the curves created by intersecting two planes with the surface.

curvatures at the point are the maximum and minimum of all possible normal curvatures [50].

In this work, we explore the use of a mesh based representation, so it is useful to convert Eq. 3.11 into its discrete counterpart. The surface integral can first be broken down into a sum of integrals over piecewise continuous regions, namely triangles. The integrals are then composed into a sum of regularly spaced sample points over the triangles, Δ, giving:

$$F(S) \approx \sum_{\{\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3\}\in\Delta} \sum_{\{\lambda_1,\lambda_2,\lambda_3\}\in A} g(\lambda_1\mathbf{v}_1 + \lambda_2\mathbf{v}_2 + \lambda_3\mathbf{v}_3, \lambda_1\mathbf{n}_1 + \lambda_2\mathbf{n}_2 + \lambda_3\mathbf{n}_3) \qquad (3.14)$$

where $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ is a triangle consisting of vertices $\mathbf{v}_1, \mathbf{v}_2$, and $\mathbf{v}_3$, each having normals $\mathbf{n}_1, \mathbf{n}_2$, and $\mathbf{n}_3$ respectively. A denotes a regular sampling over the triangle, where the sample points, $\{\lambda_1, \lambda_2, \lambda_3\}$, are barycentric coordinates satisfying $\lambda_1 + \lambda_2 + \lambda_3 = 1$ and $\lambda_i \geq 0$ for $i \in \{1, 2, 3\}$. The method of computing the error on sampling points within the triangles relates our work to other mesh based approaches [28, 105, 99, 100]. On the other hand, an alternative approach, used in the work of Duan et al. [21], is to sample the error on the tangent plane of the mesh vertices.

To recover the shape of textureless surfaces, only a small number of sample points may be necessary. In such a case, the simplest sampling scheme is to use only the vertices of the mesh as sample points. On the other hand, a sampling resolution matching the image resolution may be necessary for highly textured objects. As we would like our method to work on textured and textureless surfaces, we choose the dense sampling. The only downside of the increased sampling is a proportional increase in computation time.

Assuming a reflectance model, the shape refinement then proceeds by iteratively updating the initial shape, S_0, using Eq. 3.13 until convergence. The gradient is approximated per vertex using central differences, and the optimization is done at multiple mesh resolutions. Following Duan et al. [21], our mesh data structure is also capable of handling topological changes. A more detailed description of the implementation details are presented in Chapter 4.

3.2.3 Lambertian Object

We now consider a particular instantiation of the general shape recovery formulation presented above. Consider the simple example of shape refinement where the scene is assumed to consist of a Lambertian object. In order to do the refinement, we must first instantiate the illumination knowledge, L_i, and the rendering function, R, mentioned above. Given our assumption on known lighting, the light knowledge consists of the position and color of a point light source as well as the ambient light contribution for each input image. Given that we know the position of the source, we also know the direction to the source, $\mathbf{l}_i$, from a point. Using the Lambertian model described in Section 3.1.3, the rendering function becomes:

$$R_{lamb}(\mathbf{x}, \mathbf{n}, L_i) = \begin{pmatrix} (\mathbf{n} \cdot \mathbf{l}_i)\ell_i^r k_{d,\mathbf{x}}^r + a_i^r k_{a,\mathbf{x}}^r \\ (\mathbf{n} \cdot \mathbf{l}_i)\ell_i^g k_{d,\mathbf{x}}^g + a_i^g k_{a,\mathbf{x}}^g \\ (\mathbf{n} \cdot \mathbf{l}_i)\ell_i^b k_{d,\mathbf{x}}^b + a_i^b k_{a,\mathbf{x}}^b \end{pmatrix} \tag{3.15}$$

At any step during the optimization we need to be able to fit a Lambertian model to every surface point on the current mesh, S_t. We first assume that the ambient color of the object is the same as its diffuse color, i.e., $k_{a,\mathbf{x}}^\lambda = k_{d,\mathbf{x}}^\lambda$. For every sample point $\mathbf{x}$ on the mesh, with normal $\mathbf{n}$ (computed as the barycentric weighting of the vertex normals), every image that the point is visible gives an equation of the following form for each color channel:

$$I_i^\lambda(\Pi(\mathbf{P}_i\mathbf{x})) = ((\mathbf{n} \cdot \mathbf{l}_i)\ell_i^\lambda + a_i^\lambda)k_{d,\mathbf{x}}^\lambda \tag{3.16}$$

which is linear in the unknown albedo, $\mathbf{k}_{d,\mathbf{x}} = (k_{d,\mathbf{x}}^r, k_{d,\mathbf{x}}^g, k_{d,\mathbf{x}}^b)^T$, and can be solved in the least squares sense with a couple vector dot products and a division.

Let $V(\mathbf{x}, \mathbf{P}_i)$ be the binary function denoting the images in which $\mathbf{x}$ is visible. Setting the weight function $h(\mathbf{x}, \mathbf{n}, \mathbf{P}_i, L) = V(\mathbf{x}, \mathbf{P}_i)$, the error computed by g is essentially the residual of the fitting in Eq. 3.16:

$$g_{lamb}(\mathbf{x}, \mathbf{n}) = \sum_i V(\mathbf{x}, \mathbf{P}_i)\|I_i(\Pi(\mathbf{P}_i\mathbf{x})) - R_{lamb}(\mathbf{x}, \mathbf{n}, L_i)\|^2 \tag{3.17}$$

Image Sampling Issues

Eq. 3.17 equally uses all observations of a sample point, but this is not entirely correct. First consider the case when the current mesh is far from the actual surface, as depicted in Figure 3.5. When the current estimate is far from the surface, widely separated views sample from different regions of the underlying surface, implying that a numerical gradient may not provide meaningful information. In this case, it is helpful to restrict the number of views used in g, instead of using all visible samples. Later in the optimization process the number of views can be increased.

In order to choose a subset of the observations, we first assume that the images were captured using a stationary camera and a rotating object. Then following Esteban and Schmitt [22], we use the $n_{cameras}$ closest cameras to the median visible camera. The median visible camera is currently

based solely on the azimuthal angle (y axis is up, so azimuth is the rotation about y), allowing for multiple elevations in the capture. This modification easily fits into the above framework by modifying the visibility function to get V', which returns a binary 1 if and only if the point is visible and is within the set of $n_{cameras}$ surrounding the median camera.

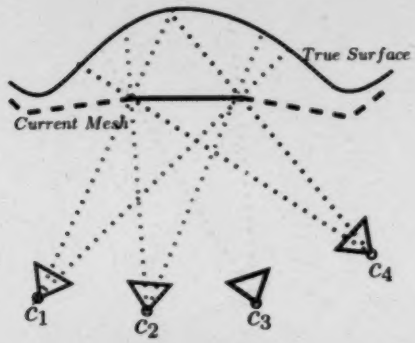

Figure 3.5: The cameras c_1 through c_4 all observe the same triangle on the current mesh. Neighboring cameras sample from overlapping regions, whereas samples from c_1 and c_4 do not overlap.

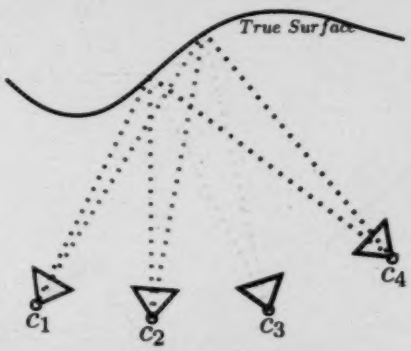

Figure 3.6: The cameras c_1 through c_4 all observe the true surface, but the number of pixels occupied by the surface patch varies between the cameras.

The other sampling issue occurs even when the true surface is known. Consider the case where several cameras observe a particular patch of the true surface (Fig. 3.6). The patch occupies a different number of pixels when projected into each of the images. In the views where the patch occupies few pixels, the sample points on the patch will project to overlapping pixels. This causes a blurred texture to appear on the patch. To account for this, each sample is weighted by $\mathbf{n} \cdot \mathbf{v}$ (in both the albedo computation and in the error computation), implying that image planes parallel to the patch are given large weights, and grazing views are given smaller weights.

This weighting is a simplification of the one used by Isidoro and Sclaroff [37], except we do not take into account the distance of the patch from the camera. This is a valid assumption, as in most of our sequences, the object is rotating in front of the camera, so the cameras are typically a constant distance from the scene. This distance is typically larger than the size of the object so the intra-object distance is assumed to have negligible effects.

We note that this second sampling issue is a consequence of our error function being computed on the surface of the object. This problem can be avoided if the error is formulated in image space.

Again, the above mentioned weight is easily embedded in the function h. Letting

$$h_w(\mathbf{x}, \mathbf{n}, \mathbf{P}_i) = \mathbf{n} \cdot \mathbf{v}$$

where $\mathbf{v}$ is derived from $\mathbf{P}_i$ and $\mathbf{x}$, we get the new instantiation of h:

$$h(\mathbf{x}, \mathbf{n}, \mathbf{P}_i, L) = h_w(\mathbf{x}, \mathbf{n}, \mathbf{P}_i) V'(\mathbf{x}, \mathbf{P}_i) = (\mathbf{n} \cdot \mathbf{v}) V'(\mathbf{x}, \mathbf{P}_i)$$

The complete Lambertian error function becomes:

$$g_{lamb}(\mathbf{x}, \mathbf{n}) = \sum_i (\mathbf{n} \cdot \mathbf{v}) V'(\mathbf{x}, \mathbf{P}_i) \| I_i(\Pi(\mathbf{P}_i \mathbf{x})) - R_{lamb}(\mathbf{x}, \mathbf{n}, L_i) \|^2 \tag{3.18}$$

For a point $\mathbf{x}$ on the current mesh, the albedo that minimizes the weighted least squares problem in Eq. 3.18 is obtained as:

$$k^{\lambda}_{d,\mathbf{x}} = \frac{\sum_i (\mathbf{n} \cdot \mathbf{v}) V'(\mathbf{x}, \mathbf{P}_i)((\mathbf{n} \cdot \mathbf{l}_i)\ell_i^{\lambda} + a_i^{\lambda}) I_i^{\lambda}(\Pi(\mathbf{P}_i \mathbf{x}))}{\sum_i (\mathbf{n} \cdot \mathbf{v}) V'(\mathbf{x}, \mathbf{P}_i)((\mathbf{n} \cdot \mathbf{l}_i)\ell_i^{\lambda} + a_i^{\lambda})^2} \tag{3.19}$$

3.2.4 Specular Object

As mentioned in Section 3.1.4, it is not always possible to accurately estimate the reflectance parameters at all points on the surface. Fortunately, this problem does not affect the usefulness of the photo-consistency measure in Eq. 3.12. This simply states that parameter estimates for some points may not reflect the true properties of the surface at this point. The fitted reflectance model does however minimize Eq. 3.12. The proper recovery (interpolation/clustering) of specular parameters over these regions need only be done when the shape optimization is complete, whereas a direct non-linear method could be used to estimate the reflectance model during the optimization of the surface. A more complete approach would be to actually use the interpolation/clustering method whenever the optimization needed to fit a reflectance model.

In practice, it is inefficient to fit a full reflectance model to each surface point during the optimization. Instead of fitting the full reflectance model, we choose to filter out the specular highlights during the optimization.

Filtering Specular Highlights

Specular highlights can be filtered out with the use of the h function. So far we have used this function to explicitly represent visibility and resolve the image sampling issues, but it can also be used to handle specular highlights. One method is to give observations where specular highlights are expected a smaller weight. This approach relies on the current estimate of $\mathbf{n}$ and would give samples having a large $\mathbf{n} \cdot \mathbf{h}$ a smaller weighting (a similar approach was taken by Marschner in the estimation of a diffuse texture map given known geometry [58]).

Another approach, and the one used in this work, is to use the fact that specular highlights typically cause a bright image observation. In this approach, the specular highlights are assumed to produce the brightest image observations on a surface point. So the n_{spec} samples with the largest intensity are not used in computation of the albedo, nor are they used in the computation of g for a point. The rest of the optimization remains the same as the Lambertian case presented in Section 3.2.3. This type of filtering is essentially another binary function, like the visibility function V. The modifications of Eq. 3.18 and Eq. 3.19 to include the filtering are straightforward and can be

accomplished by substituting the V' with a V'' that is the product of V' with the binary specular filtering function.

The number of filtered samples, n_{spec} , must be chosen such that $n_{cameras} - n_{spec} \geq 2$, so that value of g does not vanish everywhere. In practice, we ensure that the separation is greater than 2 by taking $n_{cameras} \geq 6$ and $n_{spec} = \min(n_{cameras}/3, 4)$. Although this photo-consistency measure ignores the brightest samples, we still expect the filtering to work with objects that are strictly Lambertian. As for specular objects, this filtering method should lose effectiveness when the specular component is observed in more than n_{spec} images.

3.3 Discussion

As this work follows the variational formulation of multi-view stereo, it is closely related to the other variational approaches [21, 25, 83, 41, 40, 39, 38, 98, 84]. In particular, we followed the original work of Faugeras and Keriven [25] to formulate and obtain a PDE for a general error measure. We then use a simplified PDE for the evolution. The optimization is carried out with a mesh based representation, so the method is also closely related to the other mesh based works [28, 105, 37, 99].

A specific implementation of the general error measure is then proposed for Lambertian objects. The error function is essentially the same as the one used by Jin et al. [39], except that the proposed method uses the same error measure over the entire surface, in addition to allowing for a spatially varying albedo. Assuming that the input images contain some light variation makes this possible. Similar to other existing methods [92, 105], this light variation can be obtained by moving the object with respect to the source.

A simple extension of the Lambertian method to aid the reconstruction of specular objects was then proposed. Other variational approaches have taken specularities into account (i.e., the measure of Jin et al. [41] mentioned in Section 2.5), but their methods do not directly extend to our error measure. In our method, the brightest observations of a surface point are assumed to be caused by specular reflection and are not used in the computation of the Lambertian error measure. With enough image observations, this extension should allow for the reconstruction of both Lambertian and non-Lambertian objects.

Chapter 4

System

As the previous chapter dealt with theoretical aspects regarding the shape refinement, we now turn our attention to the implementation details. Here we will discuss the individual modules that make up a complete system for outputting 3D models and surface reflectance parameters given a set of input images.

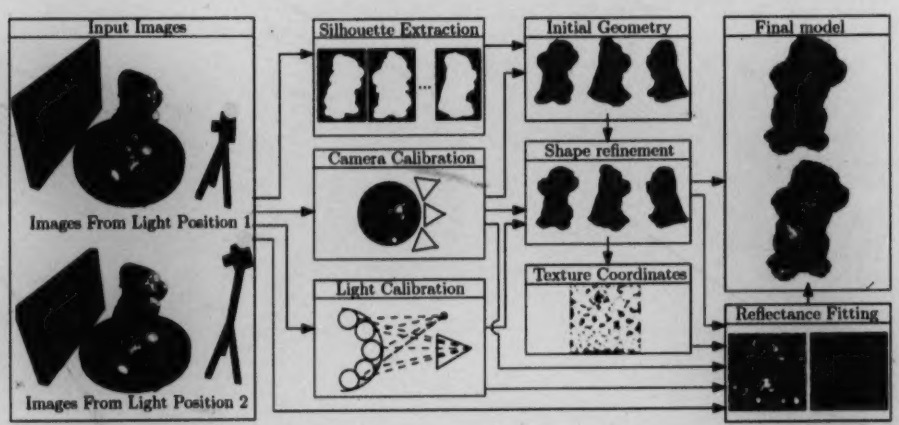

Figure 4.1: An overview of the system implementation showing the flow of information through several modules in the system.

The flow of the input images and intermediate data through the system is depicted in Fig. 4.1. We begin by discussing the prerequisites to the mesh refinement: the capture setup (Section 4.1), camera calibration (Section 4.2), light calibration (Section 4.3), and the method for obtaining an initial mesh using shape from silhouette (Section 4.4). This naturally leads to the details of the mesh based refinement (Section 4.5). Following the mesh refinement details, we discuss the implementation used for automatic texture coordinate generation (Section 4.6). Finally, we present the details of the reflectance fitting (Section 4.7). Before continuing, we remind the reader that the system itself has not only been designed as a testbed for the mesh based refinement, but also as an easy to use system with minimal hardware requirements.

4.1 Capture Setup

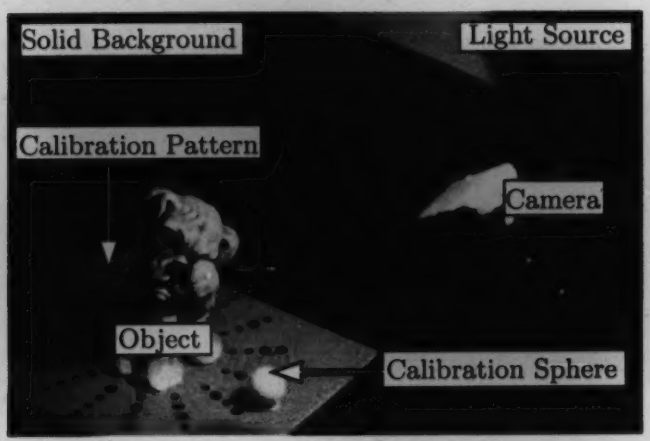

Figure 4.2: A snapshot of the capture setup.

Our capture setup consists of a single camera viewing an object rotating on a turntable (Fig. 4.2). The object is lit by a single light source, which is assumed to be contributing the majority of the light to the scene. In order to calibrate the camera, we use a calibration pattern that is situated between the object and the turntable. Alternatively, features on the object itself could be used to calibrate the cameras (see references [71, 67] for examples). The silhouette extraction is typically performed by background subtraction or blue-screening. In our implementation, we use the latter approach with a uniform colored background to obtain the silhouettes for the visual hull computation. The light position and color are calibrated using a white sphere, which rotates along with the object on the turntable.

A typical capture consists of two sets of input images, with each set containing roughly 30 input images taken as the object performs one full rotation. The two sets of input images are taken with a different light source position. For this reason, our capture setup is similar to the one described by Weber et al. [92]. As we do not account for shadowing, we currently try to place the light in a manner to avoid shadows (i.e., we try to position the light as close to the camera center as possible), implying that each rotation also has a different camera position.

The current system obeys the minimal hardware requirements, where a commonly available desk lamp, either halogen or tungsten, was used as the light source (costing roughly $20 CAD), and a painted ping-pong ball or marble was used to calibrate the light source (costing roughly $1 CAD). In many of the captures an actual turntable was used, but successful captures were also obtained by using a rotating cake table (or lazy susan), costing approximately $20 CAD. The total cost of the system would then be dominated by the camera used, where pricing for a cheap webcam or digital camera would start at $100 CAD.

4.2 Camera Calibration

To obtain the calibration parameters, all that is required is a set of point correspondences from image points to known world points. For this purpose, we use a calibration pattern based on the one used in Canons 3D S.O.M. [3]. Their pattern has the advantage of being easy to detect, stable through occlusions, as well as reliable through a wide variety of viewing angles.

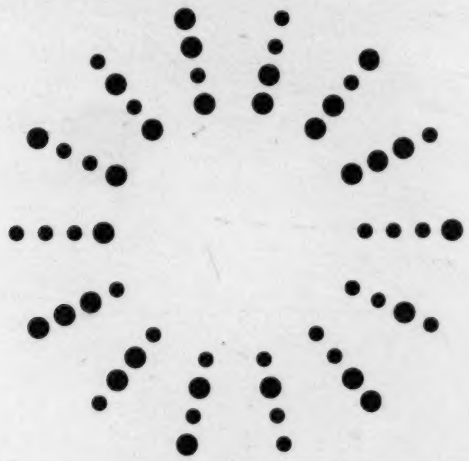

Figure 4.3: Our slightly modified version of the 3D S.O.M. pattern.

The pattern consists of 14 groups of small and big dots, where the dots are binary digits (small being binary 0, and big being binary 1), which uniquely identify the group (Fig. 4.3). The original pattern consisted of 15 possible groupings, all possible groups except the group of all small dots. We also removed the group of big dots, for reasons that are described below.

The detection of the modified pattern proceeds in the same manner as the original pattern [3], which we review here for completeness:

- *Detect Potential Dots:* A simple thresholding can be used to create a binary image, where connected components are identified as the potential dots. The sizes (in pixels) and the centers of the dots are all that is required for the remainder of detection.

- *Find Potential Groups:* Potential groups of four dots are found such that the dots lie on the same line and satisfy the cross ratio of the world points.

- *Order Dots in Groups:* The dots in a group must be ordered in a consistent manner so that the binary number can be read off. This is done by finding the center of the pattern, then ordering the groups from the center out. Each line defined by a potential group is intersected with the line of every other potential group. The intersection point of each pair of lines is used to cast votes for the center point in a low resolution discretization of the image. The grid cell with the

most votes is where the center lies, and the actual center point is then refined. Any potential groups without an intersection in this grid cell are outliers and can be discarded.

- *Classify Dots in Groups:* Each dot within a group must be classified as either large or small. If a_{max} (resp. a_{min}) is the largest (resp. smallest) dot area in a group, then a dot is small if its area is less than $((\sqrt{a_{max}} + \sqrt{a_{min}})/2)^2$ and big otherwise.

The classification of dots into big or small is the primary difference of our method from the original method. The original method performed the classification based on a_{max} only, where a dot was considered small if its area was less than a fraction of a_{max}. We found that this method sometimes misclassified the group of all big dots as the group containing three big dots and one small dot. This typically happened when the pattern was viewed at a grazing angles. We found that removing the group of all big dots and using the above classification alleviated this problem.

4.2.1 Automatic Thresholding

Detection of the potential dots via thresholding typically requires a user to manually enter the threshold value, depending on the lighting conditions at the time the image was taken. In an attempt to remove, or at least reduce the amount of user interaction, we have designed an automatic technique for detecting the dots. The technique is reliable and can detect the dots over a wide range of illumination conditions.

The method was motivated by the algorithm for detecting Maximally Stable Extremal Regions (MSER) [59], which detects features that can be reliably matched over widely separated views. The motivation of our approach is that there is a range of thresholds that will produce connected components that correspond to the same dot. Therefore, we try every possible threshold on the image (assuming an 8-bit image, there are 255 potential thresholds) in succession, while keeping track of the centers and sizes of the connected components for each threshold. If there exists a center in roughly the same position for a successive range of thresholds, then this position is extracted as a dot, and its area is taken to be the average area of the component, over the range of thresholds.

Fortunately, the implementation can be handled in a similar manner as that for detecting MSER's [59]. First, each (x, y) position in the image is sorted based on the intensity at that point. Note that this step is linear in the number of pixels, since the pixels are just placed in an array with other pixels having the same intensity. We then initialize a set of connected components to nil. Each connected component will maintain certain characteristics, such as its size, the sum of all the pixel coordinates in the component (from which the current position can be deduced), and the previous sizes and positions of the component.

A 2D array of pointers to existing components is created and cleared, and the pixels with the lowest intensity value (since we wish to detect black dots) are taken from the unprocessed set of pixels. For each pixel, the neighboring positions in the 2D array are checked. One of two possible

cases can occur:

1. *None of the neighboring pixels are set* This pixel is starting a new connected component. Create a new connected component, and set the corresponding pointer to point to this component.

2. *One or more neighboring pixel(s) is set* If the neighboring pixels belong to the same component, then this pixel is a growing part of an existing connected component. Add this pixel to the adjacent component, and increase the size. Otherwise, this pixel is connecting two or more existing components. Merge the smaller component(s) into the larger component, and adjust the sizes accordingly. Add this pixel to the merged component. Set the pixel's pointer to the component the pixel was added.

After adding all the pixels of a given intensity, for which the temporary image corresponds to a thresholded image using that intensity, we check to see if the connected components have moved. For each component, we check its current position (obtained by dividing the sum by the number of points) to the previous position of the component (if one exists), and if the component has moved less than one pixel, we increment its *stationary* counter, otherwise the *stationary* counter is cleared. If the counter exceeds a threshold τ_{min}, then we consider the component to be a dot, compute its average position and area, and store these characteristics in an output array. If on future iterations the dot has not moved, then its position and area are updated in the output array. This output position will be updated until the *stationary* counter exceeds τ_{max}. Otherwise, the component is assumed to be either too large or no longer growing, and consequently, its position is no longer updated. We have found values of $\tau_{min} = 20$ and $\tau_{max} = 100$ to be good choices. This means that a component has to stay in approximately the same position for 20 different thresholds to be chosen as a valid dot. If the component is approximately stationary for more than 100 different thresholds, the component's position and size in the output buffer will no longer be updated (see Algorithm 1).

After detection of the pattern, existing calibration methods, such as Zhang's method [107], can be applied to obtain the camera parameters. The intrinsic parameters are calibrated in advance, so during the capture we only need to obtain the extrinsic parameters. At this point in the capture we remove the radial distortion using the model described in Chapter 3.

Algorithm 1: Extract dots

```
Data: Image im
Result: VectorOfDots output
VectorOfPixels bins[256];
for y=0;y < im.h;y++ do
    for x=0;x < im.w;x++ do
        bins[im[y][x]].append ((x,y)^T);
    end
end
Component labels[im.h][im.w];
VectorOfComponents components = ∅;
for intensity ∈ [0, 255] do
    foreach p ∈ bins[intensity] do
        neigh = getNeighbors (labels,p);
        Component c;
        if ||neigh||==0 then
            c = new Component ();
            components.append (c);
        else
            //This pixel is connected to an existing component
            //or it is joining two or more components
            c=mergeComponents (labels,neigh,p);
        end
        c.sum+=p;
        c.size++;
        labels[p.y][p.x]=c;
    end
    //Have the components moved since last gray level
    foreach c ∈ components do
        if ||c.sum/c.size − c.lastPosition|| <1 then
            c.stationary++;
            if c.stationary≥ τ_min && c.stationary≤ τ_max then
                if c ∈ output then output[c]=<c.sum/c.size,c.size> ;
                else output .append (<c.sum/c.size,c.size>);
            end
        else
            c.stationary=0;
        end
        c.lastPosition=c.sum/c.size;
    end
end
```

4.3 Light Calibration

We use a single, glossy white sphere to calibrate the light source position/color. The method is similar to other works using multiple spheres to triangulate the position of light sources/features [64, 73, 52], with the main difference being that we use several temporal images of one physical sphere instead of several physical spheres.

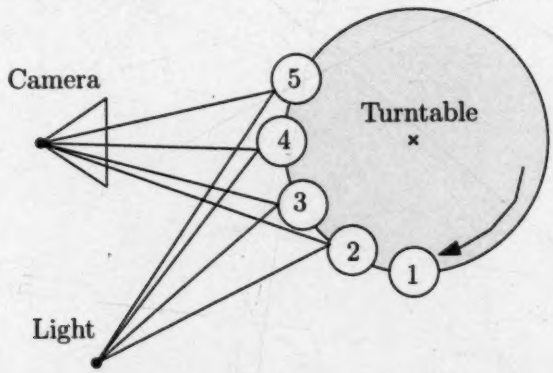

Figure 4.4: The specular highlights of a white sphere rotating on the turntable are used to find the light position.

The basic idea in all the methods is similar: an observed specular highlight (or feature) on a sphere corresponds to a ray in space. When the positions/radii of the spheres are known, a ray can be computed for each sphere, and the intersection of these rays corresponds to the light source (or feature) position. In our case, the light source position is fixed with respect to the camera, and the sphere is rotating on the turntable, so we can use multiple temporal images of the same sphere to calibrate the source position (Fig. 4.4). Because the sphere is moving with respect to the camera, the sphere's position is represented in the world coordinate system (i.e., the coordinates of the rotating turntable). On the other hand, since the light is stationary with respect to the camera, its position is represented in the camera coordinate system.

Our implementation of the light source detection and light source color estimation proceeds in the following manner:

- *Detect Sphere Position*
- *Compute Ray For Each Image of the Sphere*
- *Intersect Rays*
- *Compute Source Color*

4.3.1 Detect Sphere Position

To obtain an initial estimate to the position/radius of our sphere, we require the user to select the sphere in two input images. The selection consists of clicking on three points defining a circle that

best approximates the projection of the sphere (note that the projection of a sphere into an image is generally an ellipse). The centers of these circles are then used to triangulate the center of the sphere, $\mathbf{x}_{sphere}$ (using the algorithm given in [33]). The radius of the sphere, r_{sphere}, is determined by backprojecting a single point on the circle to the same depth as $\mathbf{x}_{sphere}$ and computing the distance of this backprojected point to $\mathbf{x}_{sphere}$.

The above procedure serves as an initial estimate of the sphere's position and radius, but the accuracy of the light source position relies on an accurate estimate of these values. To increase the accuracy, we then refine the position of the sphere based on image observations in all the input images that observe the entire sphere. We use a method inspired by Knossow et al. [42], who optimize the parameters of a kinematic model of a human so that the projected apparent contours of the model agree with the edges observed in the images. In our case, the parameters are the sphere's position and radius, and we wish to have the silhouette of the sphere agree with image edges.

The refinement reduces to a non-linear optimization of the following cost function:

$$E_{sphere} = \sum_{i \in V} \sum_{\mathbf{x} \in A_i} \min_{e \in E_i} (\|\Pi(\mathbf{P}_i \mathbf{x}) - e\|^2) \tag{4.1}$$

where V is the set of images in which the sphere is visible, A_i is a set of sample points on the apparent contour of the sphere in image i, and E_i is the set of edge locations detected in image i.

Following Knossow et al. [42], the cost in Eq. 4.1 simplifies to:

$$E_{sphere} = \sum_{i \in V} \sum_{\mathbf{x} \in A_i} d_i(\Pi(\mathbf{P}_i \mathbf{x}))^2 \tag{4.2}$$

where d_i is the distance transform of the edge features in image i.

We use the Canny edge detector to detect edge features, giving a binary image. The distance transform of these binary feature images is then performed using the approximate algorithm of Danielsson [18]. The optimization of the sphere's radius and position is done with the Levenberg-Marquardt algorithm. For each image, we select roughly 50 points on the apparent contour to be used in the optimization (see Appendix A for details on selecting the sample points on the apparent contour).

4.3.2 Compute Ray for Each Image of the Sphere

Now that we have an accurate estimate of the sphere's position, we can compute a ray from the sphere to the light source. We use the observation of a highlight on our specular sphere to define this ray. That is, in each image we detect the brightest pixel on the sphere: $(x_b, y_b)^T$. The ray corresponding to the brightest pixel, $\vec{r}(t) = \mathbf{o}_b + t\mathbf{d}_b$, is intersected with the sphere, giving a point $\mathbf{x}_{isect}$ and a surface normal $\mathbf{n}_{isect}$ at the point of intersection. The desired ray from the sphere to the light source has origin $\mathbf{x}_{isect}$, and the direction is obtained by reflecting $\mathbf{d}_b$ about $\mathbf{n}_{isect}$ (see Fig. 4.5). In practice, due to image noise, we do not use the brightest pixel on the sphere in the input image but on a Gaussian filtered image instead.

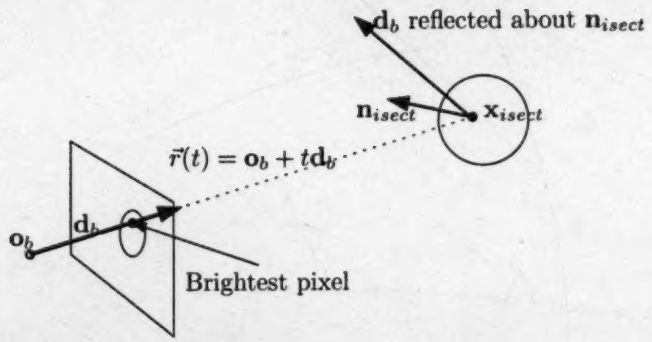

Figure 4.5: The brightest pixel on the sphere in an image defines a ray, $\vec{r}(t) = \mathbf{o}_b + t\mathbf{d}_b$, that is intersected with the sphere to find the direction to the light source.

Although we currently use the above method, which computes the ray from the specularity to the light source position, we have also explored other possibilities. For instance, by using the shading of a Lambertian sphere we can compute the direction of an infinite source. The ray from the sphere to the light source is then defined by the center of the sphere and the recovered direction. This basically assumes that the light source can be well approximated as an infinite source over the surface of the sphere, but the direction to the source changes over different positions of the sphere with respect to the camera. While this method also works, we found that using the specular highlights provided a better estimate of the light position, which is essential in the reflectance estimation (Section 4.7).

4.3.3 Intersect Rays

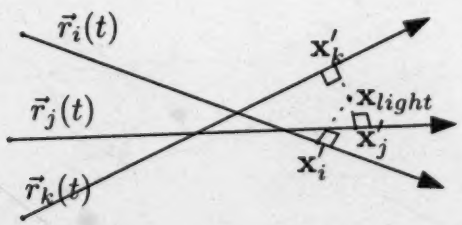

Figure 4.6: The optimization seeks to minimize the squared distance of $\mathbf{x}_{light}$ to each of the 3D rays.

The rays for each image where the sphere is visible can now be intersected to compute the light source position. Ideally the rays would all intersect in a single point, but this is rarely the case due to noise and the source having a finite area.

We obtain an initial estimate to the source position by solving each pair of equations, where $i \neq j$:

$$\mathbf{o}_i + t_i\mathbf{d}_i = \mathbf{o}_j + t_j\mathbf{d}_j \tag{4.3}$$

and then taking the average position as the initial estimate.

This above estimate is then refined using the following non-linear cost function:

$$E_{light}(\mathbf{x}_{light}) = \sum_{i \in V} \|\mathbf{x}_{light} - \mathbf{x}'_i\|^2 \tag{4.4}$$

where $\mathbf{x}'_i$ is the projection of $\mathbf{x}_{light}$ onto the ray i, given as:

$$\mathbf{x}'_i = \mathbf{o}_i + ((\mathbf{x}_{light} - \mathbf{o}_i) \cdot \mathbf{d}_i)\mathbf{d}_i$$

which is illustrated in Fig. 4.6.

The above procedure provides an accurate estimate of the light source position, but we acknowledge that the non-linear cost function is not optimal. The cost function should instead optimize the distance between the observed highlight of the source on the sphere and the synthesized highlight on the sphere.

4.3.4 Compute Source Color

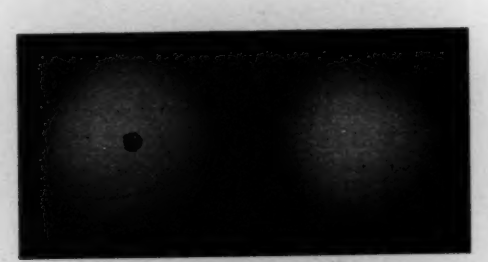

Figure 4.7: An input image of a ping-pong ball (left) and the rendered sphere after fitting the sphere and light color (right). The blueish region on the left image denotes which pixels were filtered out as specular highlights. Because the ping-pong ball is modeled as a perfect sphere, the hemispherical inconsistency in the left image is not observed in the rendered image.

Given the estimate of the sphere and the light position, we can now estimate the light source color. We use the assumption that the color of our calibration sphere is white (or is of some known reference color), meaning $\mathbf{k}_d = (1,1,1)^T$ and $\mathbf{k}_a = (1,1,1)^T$. We then filter out the anticipated specular highlight (which was used to find the light source position), and for all pixels that observe the sphere (in every image), we compute the normal $\mathbf{n}$ and the direction to the light source $\mathbf{l}$. Each pixel then gives an equation for each color channel:

$$\begin{aligned} (\mathbf{n} \cdot \mathbf{l})\ell^r + a^r &= I^r \\ (\mathbf{n} \cdot \mathbf{l})\ell^g + a^g &= I^g \\ (\mathbf{n} \cdot \mathbf{l})\ell^b + a^b &= I^b \end{aligned} \tag{4.5}$$

The linear system of equations is then solved for each color channel to obtain the light source color $\boldsymbol{\ell} = (\ell^r, \ell^g, \ell^b)^T$ and the ambient color $\mathbf{a} = (a^r, a^g, a^b)$. Given that these variables should not be negative, we use a non-negative linear least squares method to find the solution. In addition to ignoring the specular pixels, we also ignore saturated pixels. Furthermore, we only use observations

that satisfy $\mathbf{n} \cdot \mathbf{l} > 0.1$ (i.e., the point is lit by the source direction). Letting $\mathbf{h}$ be the half-angle for a particular pixel, the highlights are filtered out using the following condition:

$$\mathbf{n} \cdot \mathbf{h} \geq 0.995 \approx \cos(6^\circ) \tag{4.6}$$

The above implies that the specularity is only observed in a 6 degree cone about the perfect reflection angle, but the threshold should be adjusted if the reflectance of the calibration sphere deviates from this. In practice, the number of potential samples can be large, so we limit the number of the samples to some reasonable number, where currently we use at most 1000 samples. Figure 4.7 illustrates the results of the light color fitting.

4.4 Shape from Silhouette

We use a color based segmentation to separate the object from a solid colored background. Through this process we obtain the silhouette images, S_i, which are used as input to shape from silhouette.

4.4.1 Silhouette Images

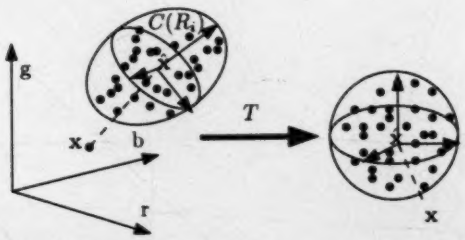

Figure 4.8: A color $\mathbf{x}$ is part of the background if it lies within a certain distance of the ellipsoid defined by the background samples. The distance is computed by transforming the point into a space where the ellipsoid is a sphere.

The user is prompted to select a region R_i containing samples of background pixel colors. Let $C(R_i)$ denote the colors extracted from the region R_i in matrix format, so $C(R_i)$ is a $3 \times m$ matrix. These background samples can be summarized by an ellipsoid in RGB color space, which can be obtained through a principle component analysis. We then wish to classify other image pixels as belonging to this set of background pixels based on their distance to this ellipsoid (see Fig. 4.8).

Letting $\hat{\mathbf{x}} = mean(C(R_i))$, and $\hat{C}(R_i)$ be the result of subtracting the mean from each column of $C(R_i)$. The principle components are obtained through the eigenvector decomposition of the covariance matrix:

$$\frac{\hat{C}(R_i)\hat{C}(R_i)^T}{(m-1)} \tag{4.7}$$

giving a 3×3 matrix $\mathbf{A}$ containing the eigenvectors in the columns, and a 3×3 matrix $\mathbf{B}$ whose diagonal is the eigenvalues. We transform the other colors, $\mathbf{x}$, into a new space using the following

transformation:

$$T(\mathbf{x}) = (\mathbf{AB}^{0.5})^{-1}(\mathbf{x} - \hat{\mathbf{x}}) \tag{4.8}$$

The length of the color vector in this new space measures the similarity of the sample $\mathbf{x}$ and the background samples. This transform takes the general ellipsoid to a sphere centered at the origin. A threshold on the length of the vector in this space, set to $4\sqrt{3}$, can then be used to classify the pixel as foreground or background. This gives a binary mask for each image, where a 1 (resp. 0) denotes that the pixel is foreground (resp. background).

In practice, multiple regions need to be selected to identify the background regions. In this case, we take the silhouette mask to be the binary AND of the individual masks.

4.4.2 Marching Intersections Shape from Silhouette

We use the marching intersections (MI) method [87] to compute the approximate visual hull from the binary image masks S_i. This method was briefly outlined in Chapter 2.

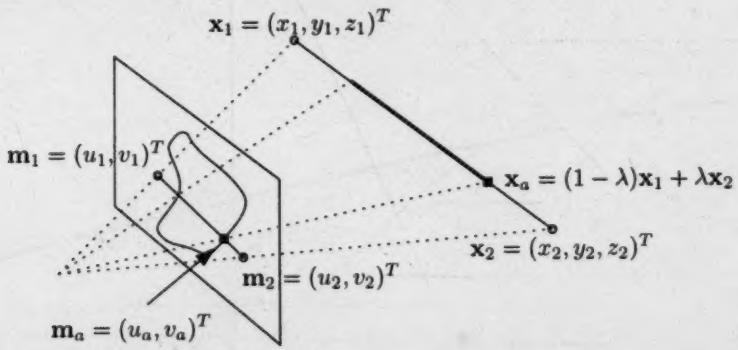
$\mathbf{x}_1 = (x_1, y_1, z_1)^T$
$\mathbf{m}_1 = (u_1, v_1)^T$
$\mathbf{x}_a = (1-\lambda)\mathbf{x}_1 + \lambda\mathbf{x}_2$
$\mathbf{x}_2 = (x_2, y_2, z_2)^T$
$\mathbf{m}_2 = (u_2, v_2)^T$
$\mathbf{m}_a = (u_a, v_a)^T$

Figure 4.9: Computing the intersection point on the 3D ray, given the intersection point in an image.

Recall that the MI data structure consists of three sets of rays that form a 3D lattice, with each ray storing entry and exit points of the true object. The entry/exit points represent a union of disjoint intervals along the ray that belong to the object. As the points along the ray can be parameterized by a linear combination of the two end points, these intervals lie in the real number line, i.e., $\bigcup_i [a_i, b_i]$, and so the entry/exit points can be stored as a list of floating point numbers. For any ray, each image gives a set of entry/exit points corresponding to a union of valid intervals. Combining the results from multiple images is identical to an intersection of the intervals for each image.

To obtain the intersection points for a particular input image, the end points of a ray are first projected onto the image, giving a line. The line is then scan-converted, checking for intersections with the silhouette. Once an intersection is found, the corresponding position on the 3D ray can be found in the following manner. Let $\mathbf{x}_1 = (x_1, y_1, z_1)^T$ and $\mathbf{x}_2 = (x_2, y_2, z_2)^T$ be the end points of a 3D ray represented in the camera coordinate frame, so that the z coordinate is the depth. Also, let $\mathbf{m}_1 = (u_1, v_1)^T$ (resp. $\mathbf{m}_2 = (u_2, v_2)^T$) be the projection of $\mathbf{x}_1$ (resp. $\mathbf{x}_2$) onto the image (Fig.

4.9). If $\mathbf{m}_a = (u_a, v_a)^T$, then the corresponding intersection point in 3D, $\mathbf{x}_a = (1-\lambda)\mathbf{x}_1 + \lambda\mathbf{x}_2$, is obtained by solving for λ as:

$$\lambda = \frac{x_1 - u_a z_1}{u_a(z_2 - z_1) - x_2 + x_1} \tag{4.9}$$

or:

$$\lambda = \frac{y_1 - v_a z_1}{v_a(z_2 - z_1) - y_2 + y_1} \tag{4.10}$$

The better conditioned of the two equations is used for the solution to λ.

The remainder of an implementation of SFS using MI is straightforward, although some optimizations are pointed by Tarini et al.[87]. As in the original work, after the set of entry/exit intervals on all rays have been computed, the marching cubes surface extraction [56] is used to obtain a triangulated mesh from the MI data structure.

4.5 Mesh Refinement

Now that all the necessary prerequisites have been defined, we can move onto the mesh refinement. We begin this section by describing the mesh implementation used, followed by details regarding the optimization of the error on this mesh.

4.5.1 Deformable Mesh

One of the most important design decisions that comes into play when choosing a mesh representation is whether the mesh resolution should be adaptive or not. The main advantage of such an implementation is obvious: a lower resolution mesh in regions of low detail implies less variables to optimize over. On the other hand, the adaptive mesh may create some difficulties when reasoning about mesh smoothness and self-collisions. We would like our mesh to handle topology changes (splitting and merging), so we opt for an easier implementation as opposed to a more elegant adaptive implementation.

We use the mesh proposed by Lachaud and Montanvert [46], which easily handles topology changes and maintains a consistent resolution based on some simple assumptions:

1. If e is an edge in the mesh, then $\sigma_{mesh} \leq \|e\| < 2.5\sigma_{mesh}$.
2. If v_1 and v_2 are vertices in the mesh that are not adjacent, then $\|v_1 - v_2\| > \frac{2.5}{\sqrt{3}}\sigma_{mesh}$.

The first assumption says that the length of every edge in the mesh is restricted to be within the given range and maintains an approximately constant global resolution. If an edge becomes too long, the edge is split; conversely, if the edge is too short, it is collapsed.

The second assumption is used to detect collisions. This assumption is also a restriction on the type of objects representable by the mesh, but for moderate or small σ_{mesh} this is not a problem. The other topology change occurs when the mesh is splitting (i.e., the inverse of a collision). The

detection of mesh splitting is really a special case of $\sigma_{mesh} \leq \|e\|$ being violated and is easily detected.

We will use the term *remesh* to refer to an operation that enforces the above assumptions on the mesh (i.e., removes small edges, changes the topology, etc.). By simply changing the value of σ_{mesh} and performing the *remesh* operation effectively changes the global resolution of the mesh. Following Yu et al. [100], we assume that the normal at a vertex is the weighted average of the normals at the adjacent triangles, where the weight of a triangle is proportional to its area.

4.5.2 Error Optimization

As mentioned in Section 3.2.4, our photo-consistency function essentially assumes a Lambertian model, but accounts for the rare cases where specular highlights occur by filtering them out. The photo-consistency measure is computed at various sample points on the triangles. The motion of the surface is then guided by the gradient of this measure and a regularizing term proportional to the mean curvature.

Sample Points

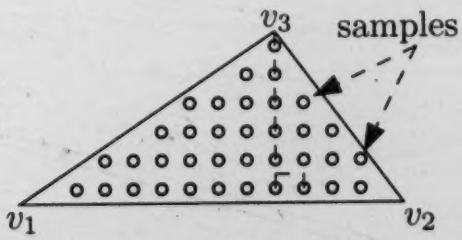

Figure 4.10: Sample points are chosen on an orthogonal lattice within each triangle.

The sample points are selected on an orthogonal lattice in each triangle of the surface (Fig. 4.10). Following other mesh implementations [28, 105], the resolution of the lattice is determined by the resolution of the images. Our cameras are approximately equidistant from the center of the object, so we define this resolution by back-projecting a single pixel from an arbitrary image onto a plane at the center of the object and parallel to the image plane. The dimensions of the back-projected pixel are used to make the lattice agree with the resolution of the input image when the plane is parallel with the image. We find using half the resolution is appropriate, as many of the triangles are not parallel to any image plane.

Given the sample points, computation of g on the mesh is performed on each sample point using the consistency function discussed in Section 3.2.3 with the specular filtering discussed in Section 3.2.4. During the optimization we need the component of the photo-consistency evaluated on a particular vertex. This per vertex consistency is computed as a barycentric weighting of the consistency on the sample points of the triangles adjacent to the vertex.

Gradient Computation

The shape evolution Equation 3.13 (presented in Section 3.2.2) used in our implementation requires the gradient of the photo-consistency function g along the direction of the normal vector. In our implementation, with a triangulated mesh, we compute $\nabla g \cdot \mathbf{n}$ numerically using central differences. Letting $g_{\mathbf{v}^+}$ (resp. $g_{\mathbf{v}^-}$) be the consistency computed on the mesh when a vertex $\mathbf{v}$ is replaced with $\mathbf{v}^+ = \mathbf{v} + \mathbf{n}\Delta\mathbf{n}$ (resp. $\mathbf{v}^- = \mathbf{v} - \mathbf{n}\Delta\mathbf{n}$), then:

$$\nabla g \cdot \mathbf{n} \approx \frac{g_{\mathbf{v}^+} - g_{\mathbf{v}^-}}{2\Delta\mathbf{n}}$$

where $\Delta\mathbf{n} = c_\Delta \sigma_{mesh}$ and $c_\Delta \in (0, 1]$, to ensure that the derivative step size is bounded by the minimum edge length σ_{mesh}.

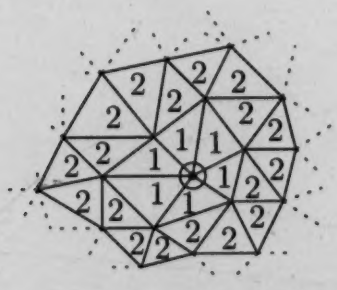

Figure 4.11: Computing the gradient at the circled vertex directly influences the adjacent triangles (marked with 1's), and influences the normals at the distance 2 neighbors (marked with 2's).

Figure 4.12: An exaggeration of the displacement at a vertex $\mathbf{v}$, demonstrating how the neighboring vertex normals are affected in the gradient computation. This in turn affects the interpolated samples over the triangles at distance 2 from $\mathbf{v}$.

In order to compute the gradient efficiently, without displacing each vertex and computing the consistency over the entire mesh, we consider the influence a single vertex has on the mesh. We will assume that displacing a vertex a small amount does not affect the visibility of the mesh, so only local effects need be considered.

When displacing a vertex $\mathbf{v}$ along its normal, the directly connected triangles are altered, and therefore their projection into the images is also altered. Also, since the normal at a vertex is computed using a weighted average of the triangle normals, the normals at the neighboring vertices are also affected. All triangles within distance two are affected (Fig. 4.11) because the photo-consistency is also dependent on the normal. The sampling points on the adjacent triangles (those marked with a 1 in Fig. 4.11) may change, as does the normal at the sample points within these triangles. For the triangles at a distance 2 (those marked with a 2 in Fig. 4.11), the interpolated normal at the sample points changes (Fig. 4.12), but the projection of these points into the images remains fixed.

Under this reasoning, the gradient computation for a vertex $\mathbf{v}$ must do the albedo fitting and consistency computation for all triangles within distance 2 of the vertex. Though this can be done

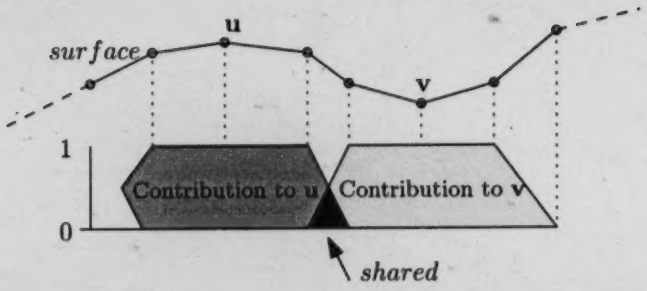

Figure 4.13: An example of the gradient computation by computing the consistency measure on the entire mesh. The neighboring triangles affect a single vertex, while distance 2 triangles partially influence the gradient on more than one vertex.

by individually moving the vertex and computing the consistency on the required triangles, we choose a method based on that of Zach et al. [102], which computes the gradient for a set of vertices simultaneously.

The mesh is partitioned into disjoint sets of vertices, U_i, such that

$$\forall \mathbf{u}, \mathbf{v} \in U_i, reqneigh(\mathbf{v}) \cap reqneigh(\mathbf{u}) = \emptyset$$

where $reqneigh(\mathbf{v})$ in our case is the set of triangles of distance 1 or 2 from $\mathbf{v}$. In other words, if $\mathbf{u}, \mathbf{v} \in U_i$ then 4 edges must be crossed to get from $\mathbf{u}$ to $\mathbf{v}$. The gradient for every vertex in U_i can be computed by simultaneously moving each vertex in U_i and computing the consistency measure over the entire mesh. The change in the photo-consistency measure for a vertex $\mathbf{v}$ then needs to be accumulated over the triangles in $reqneigh(\mathbf{v})$, instead of over the entire mesh.

The majority of time in our implementation is spent computing the gradient. As the gradient is then dependent on the number of disjoint sets used in the gradient computation, one way to improve performance is to reduce the number of disjoint sets. We first make the assumption that the photo-consistency on the triangles at distance two is expected to change less than the adjacent triangles. Then we can reduce the number of sets by reducing the separation of vertices in the sets from 4 edges to 3. This implies that two vertices $\mathbf{u}, \mathbf{v} \in U_i$ may have a non-empty $reqneigh$ intersection, where the intersection contains triangles at distance 2 from both $\mathbf{u}$ and $\mathbf{v}$. Then instead of accumulating the gradient over all triangles in $reqneigh(\mathbf{v})$, we compute the gradient at $\mathbf{v}$ as the sum of the change of consistency on $\mathbf{v}$ plus the change on the neighboring vertices. Recall that the photo-consistency at a vertex is computed as a barycentric weighting of the photo-consistency at all sample points on triangles that contain the vertex. Therefore, this faster gradient computation accurately accounts for the change on the adjacent triangles and approximately accounts for the change in photo-consistency on the distance 2 triangles (Fig. 4.13).

Regularization

The motion along the gradient is only one part of the shape evolution equation. We also need to compute the portion of the motion corresponding to the smoothing term, i.e., the motion along the normal proportional to $g\kappa$. The computation of g was previously discussed, reducing the discussion to the computation of κ, the mean curvature of the mesh.

We use the results of Surazhsky et al. [85], who survey several methods for computing mean and Gaussian curvature on triangulated meshes, suggesting that a paraboloid method provides good results for mean curvature. The method first computes a transformation aligning the normal at a vertex $\mathbf{v}$ to the $z - axis$, and placing vertex $\mathbf{v}$ at the origin. The neighbors of a vertex $\mathbf{v}$ are also transformed by this transformation. Letting (x_i, y_i, z_i) be the transformed points, a paraboloid described by parameters a, b, and c is then fit in the least squares sense to the sample points:

$$argmin_{a,b,c} \sum_i \|z_i - (ax_i^2 + bx_iy_i + cy_i^2)\|^2$$

The mean curvature at vertex $\mathbf{v}$ is then computed as $\kappa = a + c$.

4.5.3 The Shape Optimization Algorithm

As we have discussed all the components that make up the specifics of the implementation, we can now describe the complete algorithm. The pseudocode for the shape optimization algorithm is given in Algorithm 2 and will be outlined below.

The algorithm initializes the shape using shape from silhouette (line 1) and starts at some user defined base resolution (line 2). The shape is then partitioned, the sample points are obtained, the visibility is set, and the best cameras for the sample points are initialized (lines 5-8). For each partition, the gradient is computed by displacing the vertices in the partition up and then down, and recomputing the albedo and consistency at the displaced positions (lines 12-21). Each vertex is then moved in its normal direction as defined by the evolution (lines 22-24). After the vertices are moved, the shape is then remeshed (line 25).

In theory, the error (i.e., total photo-consistency) should not increase as long as the time step is small enough, but in practice, due to the remesh operation and the sampling of points on the triangles, the error sometimes increases between iterations. Therefore, in our implementation, the refinement for a given resolution stops (i.e., the *errorDecreasing* condition in the algorithm fails) when the error has not decreased after two iterations. The optimization occurs over higher resolutions by decreasing σ_{mesh} until the highest resolution is reached. This resolution may be user defined, or the increase in resolution may continue until there is only one sample point per triangle.

In practice, the movement in the direction of the normal is limited to that used in the gradient computation (i.e., by $c_\Delta \sigma_{mesh}$), and the time step should be chosen so that this condition is met by the majority of vertices. Recall that the c_{reg} coefficient was introduced to weight the regularization

Algorithm 2: Shape Optimization

```
1  shape=initializeToSFS();
2  σ_mesh=selectInitialMeshResolution;
3  while σ_mesh ≥ highestResolution do
4      while errorDecreasing do
5          U=partitionShape(shape);
6          s=getSamplePoints(shape);
7          v=getVisibility(shape,s);
8          c=getBestCameras(shape,s,v);
9          a=fitAlbedo(shape,s,v,c);
10         err=getPerVertexConsistency(shape,s,v,c,a);
11         κ=getMeanCurvature(shape);
12         foreach U_i ∈ U do
13             shapeup=displaceAlongNormal(shape,U_i,c_Δ σ_mesh);
14             albup=fitAlbedo(shapeup,s,v,c);
15             errup=getPerVertexConsistency(shapeup,s,v,c,albup);
16             shapedn=displaceAlongNormal(shape,U_i,−c_Δ σ_mesh);
17             albdn=fitAlbedo(shapedn,s,v,c);
18             errdn=getPerVertexConsistency(shapedn,s,v,c,albdn);
19             gradtemp=(errup-errdn)/(2c_Δ σ_mesh);
20             ∇g(U_i)=accumulateOverNeighbors(U_i,shape,gradtemp);
21         end
22         foreach (x, n) ∈ shape do
23             x+=(2err(x)κ(x)c_reg − ∇g(x))nΔt;
24         end
25         shape = remesh(shape);
26     end
27     σ_mesh− = Δ_mesh;
28 end
```

component, allowing user control on the smoothness of the object. In all our experiments, we found that $c_{reg} = 0.5$ produced acceptable results.

Optimizing at several mesh resolutions is important because a coarse resolution of the mesh has fewer degrees of freedom and larger changes in shape can occur. Additional improvements can be gained in regions of high frequency texture by using downsampled images during the multi-resolution search. More specifically, we construct a Gaussian pyramid of downsampled input images and choose the pyramid level that best suits the resolution of the mesh (a similar suggestion to this was made by Fua and Leclerc [28]). As the number of sample points on a triangle are chosen from the resolution of the images, the pyramid level for a given mesh resolution is chosen so that an edge of length σ_{mesh} has roughly two sample points.

Silhouette Preservation

Although the multi-resolution mesh and Gaussian pyramid scheme improves the efficiency of the refinement, a low resolution mesh is unable to accurately represent high geometric detail. In our experiments, refinement with a low resolution mesh was often unable to preserve features observed

Figure 4.14: Several iterations of a silhouette preserving displacement applied to a smoothed object.

in the silhouettes. We have found that a silhouette preserving force helped overcome these problems.

Our silhouette preserving force is similar to the other approaches proposed in the literature [24, 37, 68]. To ensure that the current mesh does not extend past the initial shape from silhouette volume, any vertex that projects outside the silhouette in a particular view is moved so that it projects to the silhouette boundary in that view. We then make the assumption that the *silhouette vertices* for each view should align with the boundary of the input silhouette image in that view. Therefore, we compute the silhouette vertices for each view (those vertices that project to the silhouette boundary of the current mesh), and compute a displacement along the normal to move these vertices closer to the input silhouette boundary for that view. The displacement is proportional to the distance of the vertex from the nearest point on the input silhouette image boundary (easily computed in a preprocess using an approximate Euclidean distance transform). If a vertex is a silhouette vertex in more than one input image, then the minimum of the displacements is retained. The displacement can then be smoothed over the vertices of the mesh to ensure a smooth model.

We then optionally apply this silhouette displacement intermittently during the optimization, where currently it is applied every 10 iterations and after an increase in resolution. Figure 4.14 shows the result of successively interleaving the silhouette preserving displacement of the mesh with the remesh operation.

4.6 Texture Coordinate Generation

To obtain a complete 3D model, suitable for today's 3D applications, we first need to obtain texture coordinates for the triangles of the mesh. In other words, we need to find a parameterization of the surface mapping a point in 2D to a point on the surface. It is in this space that the final reflectance model will be represented.

The automatic texture coordinate generation follows the method used by Lévy et al. [54]. The surface is split into pieces, which are individually mapped to 2D, and then all the pieces are placed into a rectangle. The resulting texture coordinate mapping should preserve (as best as possible) the properties of the triangles (i.e., angles), the relative size of the triangles, while utilizing as much area of the final texture rectangle as possible.

In order to split the surface, we use the shape approximation method of Cohen-Steiner et al. [15]. Their method best approximates the surface of a 3D object with a predefined number of planes (n_{planes}). A mapping from each triangle to its best approximating plane is stored during their algorithm. At the end of the approximation, we use this labeling to partition our 3D object.

Our method then proceeds in much the same way as that described by Lévy et al. [54], where the individual pieces are first mapped to 2D using a Least Squares Conformal Mapping (LSCM). The pieces are then scaled to occupy an area proportional to their surface area and packed into the rectangle starting with the largest piece first. Lévy et al. propose a packing method that attempts to minimize the wasted space between the lower contour of the current piece and the upper contour of the currently packed pieces. Our method differs slightly in this packing stage, and will be discussed below.

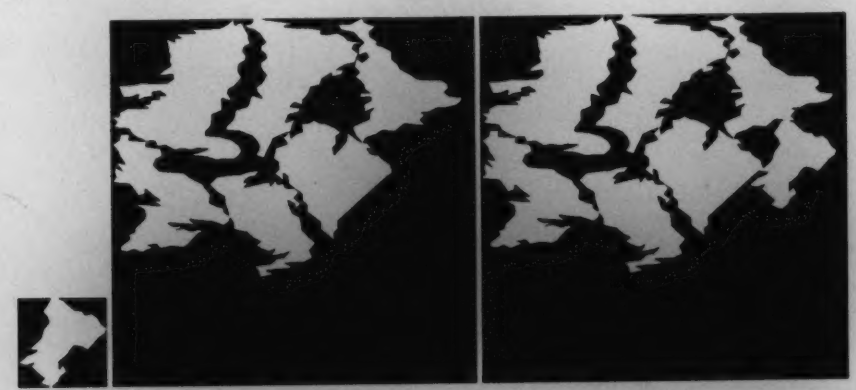

Figure 4.15: The binary mask of the piece (left) is to be inserted into the global occupancy mask (middle). The texture coordinate offset for the piece is obtained by finding the first available position for the piece, illustrated in the updated occupancy mask (right).

Given a desired resolution of the texture, say $w_{tex} \times h_{tex}$, we retain a binary mask of size $w_{tex} \times h_{tex}$ denoting which texels are used, called the global occupancy mask. This mask is initially cleared. Then each piece is rasterized to an individual occupancy mask. The texture coordinates of a piece are then computed by finding a position in the global occupancy mask that accommodates the individual mask. Actually, the search is to find a texture coordinate offset, which positions the texture coordinates of the piece in a non-overlapping part of the texture map (see Fig. 4.15). This can be performed by a binary AND of the individual mask and the global occupancy mask at each possible (u, v) location. The first position where the binary AND (across all pixels of the individual mask) returns FALSE is accepted as the texture coordinate offset for the piece. If no such position is found, then the pieces are too big to fit in the desired rectangle. In this case, the texture coordinate offsets of each piece are reset, and then each piece is scaled down uniformly. The process continues until all pieces fit in the rectangle (see Fig. 4.16 for the results of the texture coordinate generation).

The slowest part of the algorithm is checking to see if there is any overlap between an individual

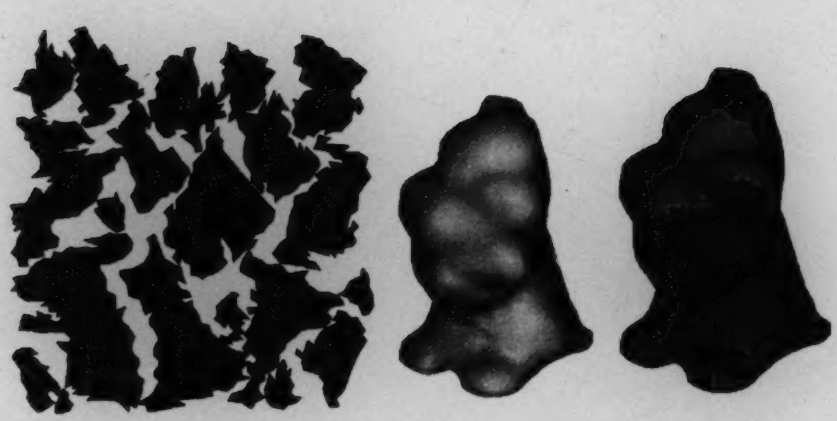

Figure 4.16: A rendering of the texture coordinates (left) for an object (middle). The image on the right is obtained by texture mapping the model using the image on the left as the texture.

mask and the global mask. This portion of the algorithm can be sped up by a fast reject implementation. For instance, one such approach is to check every n-th pixel of the mask for overlap. Once overlap is found, the position is rejected; otherwise, the remaining pixels need to be checked.

Cracks on the boundaries of the individual pieces are avoided in a similar manner as Marschner [58]. Instead of using a disjoint partitioning of the object, the partitions also include the neighboring triangles, i.e., there exists a single triangle of overlap at the boundaries of the partitions. The texture coordinate generation is done with these extra triangles in place, as is any warping of input images to the texture space. After the warping described in Section 4.7 is complete, the extra triangles are removed.

4.7 Reflectance Fitting

With the triangulated mesh and associated texture coordinates, we can now discuss the final fitting of the Phong reflectance model. As mentioned in Section 3.2.4, we did not fit an explicit specular model to the surface points because specular highlights are not necessarily observed on every surface point. If instead only a Lambertian model is required, a similar fitting process to the one defined in Section 3.2.3 can be used.

The texture coordinates provide a mapping from 2D to the surface of the mesh, so reflectance fitting takes place in each occupied texel. The center of each texel can be treated as an individual point on the object, which could then be projected into the input images to obtain color samples. Instead, we warp the input images to the texture space, using projective texture mapping and a shadow map visibility test. This approach can take advantage of the graphics hardware for trilinear filtering via mipmaps.

4.7.1 Interpolation Method

Our interpolation method is similar in spirit to the other interpolation based methods [77, 76]. We first attempt to fit the Phong model to the observations at each texel. A reliable fitting is only possible when a texel has several observations with a large $\mathbf{n} \cdot \mathbf{h}$. If there are not enough observations, the specular parameters will not be estimated correctly, leaving only a correctly fit Lambertian model.

To allow for the recovery of specular parameters on surface points where specular highlights were not observed, some sort of interpolation is required. We make the assumption that surface points with similar diffuse albedos will tend to have similar specular parameters. With this assumption we then fit the specular parameters to those texels that observe a specular highlight, giving a set of candidate Phong parameters. The remaining texels then assume the Phong parameters of one of the candidates having a similar diffuse albedo. The process is outlined below:

1. Fit a full Phong model to those texels where the specular parameters can be reliably estimated, and add the texel and its Phong parameters to the candidate set.

   - The full Phong model is only fit to those texels having at least 4 observations where $\mathbf{n} \cdot \mathbf{h} > 0.75$. As a result of the estimated shape having some errors, the surface normals are sometimes slightly wrong. Correct surface normals should have an observed specular peak in a sample where $\mathbf{n} \cdot \mathbf{h}$ is the largest. Therefore, we also reject texels whose brightest observation does not occur near the observation having the largest $\mathbf{n} \cdot \mathbf{h}$.
   - In the current implementation, we assume that the specular color is some shade of gray, i.e., $k_s^r = k_s^g = k_s^b = k_s$. We also assume that the ambient color of the object is the same as its diffuse color, so $\mathbf{k}_d = \mathbf{k}_a$. Therefore, the specular fitting process need only estimate the diffuse albedo, $\mathbf{k}_d$; the specular coefficient, k_s; and the specular exponent, n.
   - Let $\mathbf{c}_i = (c_i^r, c_i^g, c_i^b)^T$ be observed color i and let $a_i^\lambda = f_{phong}^\lambda(\mathbf{k}_d, k_s, n, i)$ be the color component anticipated by the Phong model in observation i. We then minimize the following set of non-Linear equations:

     $$E(\mathbf{k}_d, k_s, n) = \sum_i \sum_{\lambda \in \{r,g,b\}} f_{sat}(a_i^\lambda, c_i^\lambda)(a_i^\lambda - c_i^\lambda)^2 \qquad (4.11)$$

     If an observed color is saturated, the error is only considered when the anticipated color is less than the observed color. This is done through the f_{sat} function:

     $$f_{sat}(a^\lambda, c^\lambda) = \begin{cases} 0 & \text{if } c^\lambda \text{ is saturated and } a^\lambda > c^\lambda \\ 1 & \text{otherwise} \end{cases} \qquad (4.12)$$

     The use of this function was based on an observation by Schirmacher et al. [79], where ignoring the saturated pixels produced a closer fit to the unsaturated pixels.

2. For every texel not in the candidate set,

- We first fit the diffuse albedo only (in a similar manner as Section 3.2.3).
- We then find k elements from the candidate set having the most similar diffuse color (measured using Euclidean distance in YUV space). The specular parameters of this texel are chosen as the specular parameters of one of the k candidates that best agree with the image observations. In other words, these are the specular parameters that produce the least error when plugged into equation Eq. 4.11, using this texel's observations and diffuse color..

At the end of the reflectance fitting, it is possible that neighboring texels have abruptly changing specular parameters. To make synthetic renderings more visually pleasing, we perform a post-process filter to the fitted specular parameters. This filtering assigns new specular parameters to a texel based on a weighted average of the specular parameters on the texels in the same triangle and those in neighboring triangles. The weight is the sum of two terms: a term that is inversely proportional to the distance of the diffuse colors in YUV space, and a term that is inversely proportional to the geometric distance between the two points. The first term gives the parameters of similar colored texels a larger weight than differently colored texels. The second term gives closer texels a greater weight than farther texels. For all the results in this paper, the above filtering has been iteratively applied 4 times.

Unfortunately, as this method is interpolating specular parameters, it is possible that a completely Lambertian point is assigned a specular component. The clustering method of Lensch et al. [51], which uses a set of surface points to fit the specular component, should be less susceptible to this type of error.

Chapter 5

Experiments

We now discuss the results of our mesh based refinement described in Chapter 3, with sequences obtained using the system defined in Chapter 4. We first present results on synthetic sequences, and then continue to discuss the results on real experiments. In all cases, unless otherwise specified, the capture consists of two sets of images, where each set of images uses a different light position. Also, since we have not yet accounted for shadows, we try to position the light source in such a manner that there are no shadows. For real sequences, we cannot position the light source at the center of the camera, so some shadows do exist, but we expect the effect of these shadows to be minimal. Again, since the light source is close to, but not exactly at the camera center, the light source calibration presented in Chapter 4 is still used.

5.1 Synthetic Objects

Our first set of experiments are on a synthetic object, captured under similar settings as described in Chapter 4. Experiments on synthetic data sets give us an impression of how well the method works on perfect data, i.e., without image noise, calibration errors, etc. In the synthetic cases we wish to evaluate the shape evolution, and therefore we do not use the silhouette preserving force. Furthermore, all the results on synthetic objects have been obtained using the specular filtering from Section 3.2.4. Applying this filtering on Lambertian objects demonstrates that the filtering does not need to be applied selectively, but rather that it can be applied to all surfaces.

For the synthetic tests, we have chosen an object that has several different concavities that are not recovered by the visual hull. We perform various tests with this object by varying the surface texture and specular parameters. All of the data sets for the synthetic data have been obtained using two sets of 32 images taken at two different elevations, with the light source positioned exactly at the camera center, for a total of 64 images.

5.1.1 Varying Texture

The first set of results are with respect to varying the texture on the surface of the object. These experiments were intended to identify situations when the refinement will work and when it will fail. Figure 5.1 shows the sample textures used, as well as some of the input images.

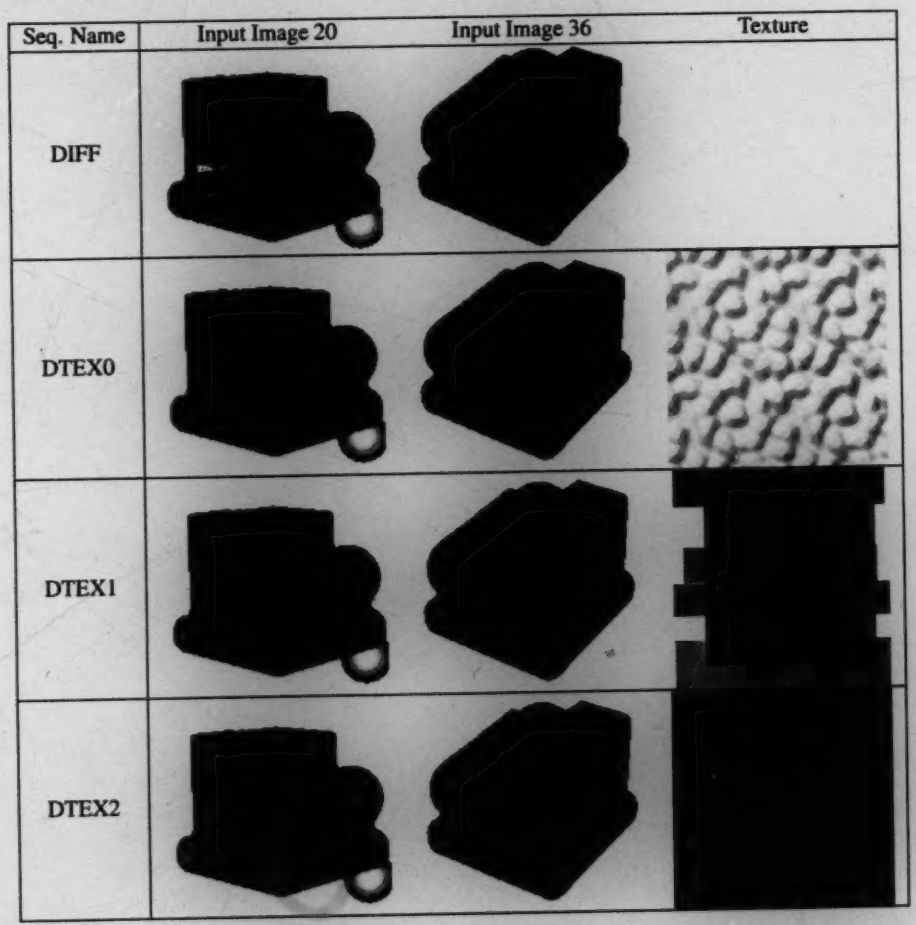

Figure 5.1: The names of the data sets and 2 of the 64 input images from each of the diffuse sequences. On the right is the texture used for the sequence.

The shape refinement was then run on each of these sequences, with $n_{cameras}$ set to 12. The starting resolution of the mesh was chosen such that the downsampled images in the Gaussian pyramid were not used. In other words, the refinement is done on the original images. The results for these experiments are shown in Figure 5.3.

Comparing the results in Figure 5.3 with the initial mesh and the ground truth (Fig. 5.2), we see that our method appears to work well on all the concavities of the object. The lighting information is sufficient to capture the shape of the object when there is no surface texture. Furthermore, the

method also appears to work well on the smoothly varying texture (DTEX0). The method is also able to recover the shape under the abruptly changing, piecewise constant texture in the DTEX1 sequence. Although, in this sequence the recovered shape has minor ridges on the boundary of colored regions in the surface texture.

Unfortunately, the results are not so impressive on the repetitive high frequency texture of sequence DTEX2. Portions of the concavities are recovered, but they are not recovered completely. This example demonstrates that the shape refinement may have trouble working on surfaces with high frequency texture. A possible explanation is that the gradient provides little information when the shape is far from the surface, making the refinement in such regions useless.

On the other hand, when we reduced the mesh resolution so that the downsampled images in the pyramid were used, we found that the method was successful on the DTEX2 sequence. This seems to suggest the importance of the Gaussian pyramid for surfaces with high frequency textures. Furthermore, it appears that the textured sequences benefit more from the multi-resolution search than the non-texture sequences, suggesting that care needs to be taken in choosing the starting resolution for a particular object.

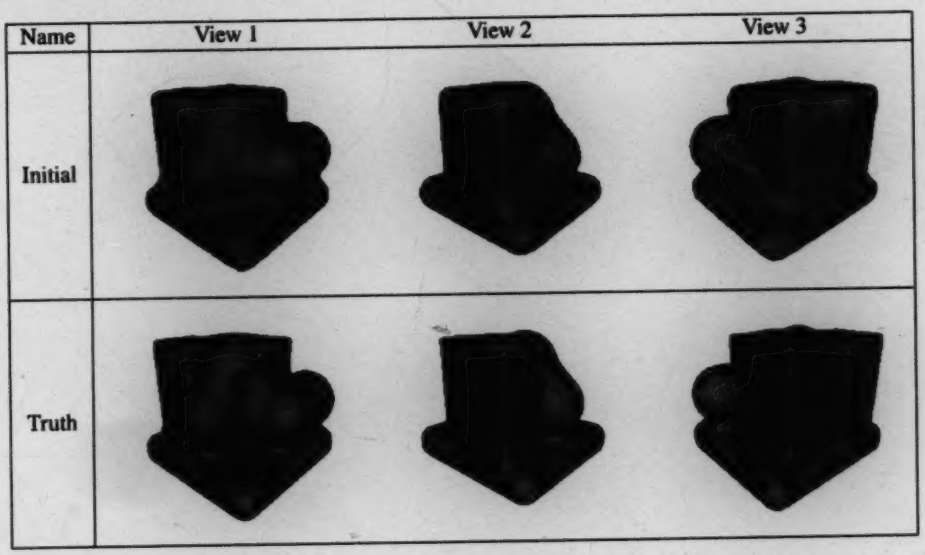

Figure 5.2: The initial shape from silhouette mesh and the ground truth of the synthetic object.

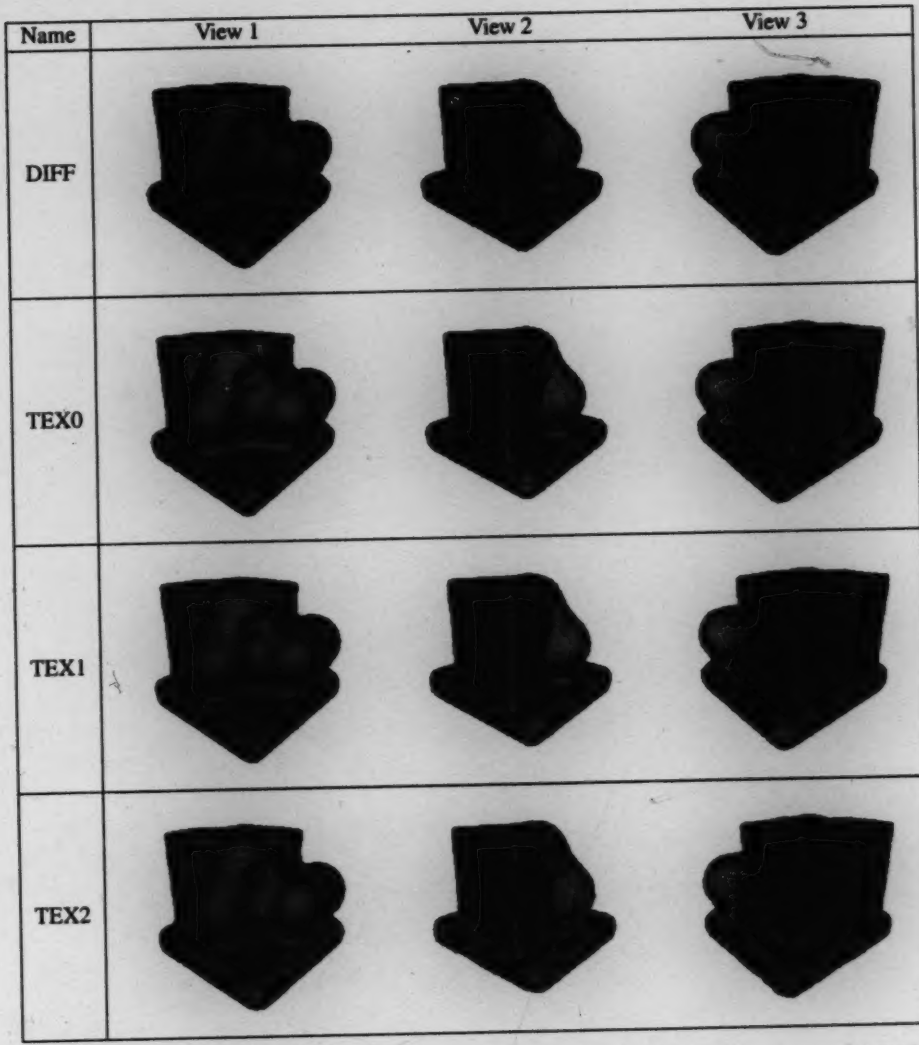

Figure 5.3: The results of the synthetic diffuse experiments. For the first three surface types (DIFF, TEX0, and TEX1), the shape was successfully recovered. The high frequency texture in the TEX2 sequence caused some difficulties for the reconstruction, which is illustrated by the remaining volume in the concavities.

5.1.2 Specular Objects

We have also performed tests when the synthetic object had a specular surface. Figure 5.4 shows some of the input images for the specular sequences. Sequence H50 was taken with a specular coefficient of 1.0 and a hardness of 50, while H20 had a specular coefficient of 0.52 and a hardness of 20. Any surface with a hardness greater than 50 will produce less highlights, implying that the results should be as good, if not better, in those circumstances.

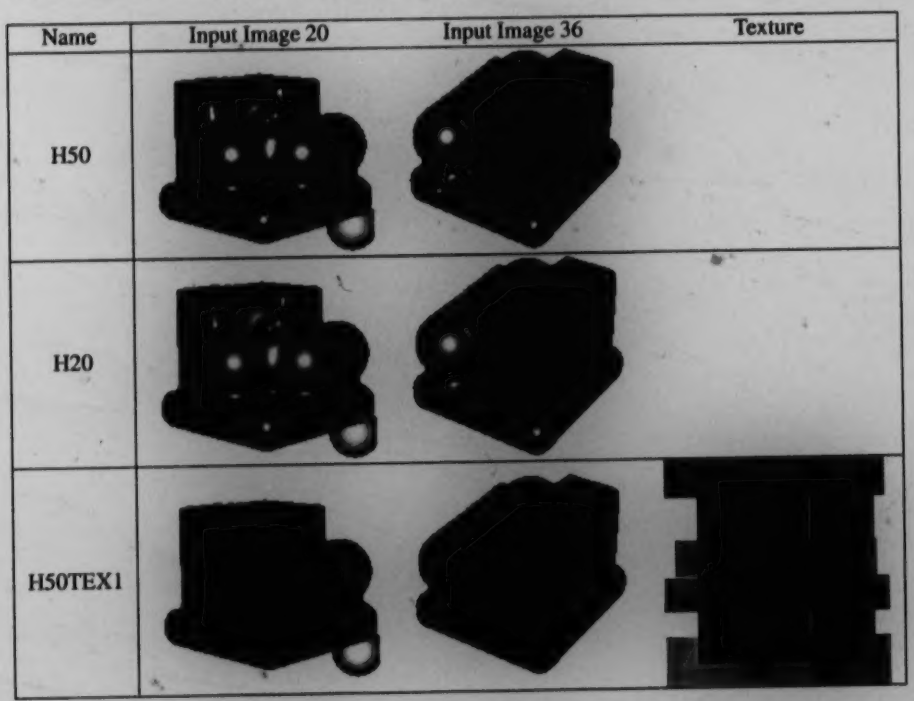

Figure 5.4: The names of the data sets and 2 of the 64 input images from each of the specular sequences. On the right is the texture used for the sequence.

As with the results in the diffuse case, we have tested the results of our method when the specular filtering was on (the brightest n_{spec} samples are not used in the reflectance fitting nor are they used in the computation of the error). We have turned off this feature so as to compare the benefits of such an approach. We have only done this for sequence H50, the results of which we call H50L.

Figure 5.5 shows the results for the specular sequences. Notice that the main concavities are recovered for both sets of specular parameters and uniform material (H50 and H20). The method also works for the textured sequence as depicted by the results for H50TEX1. Looking at the results for H50L, we see that the specular highlights cause the recovered shape to be deformed. This deformation happens on the regions where the specular highlights were observed in the image sequence. Comparing H50L to H50, we see that the specular filtering approach does indeed improve the results. Returning back to H20, the filtering is proving to be less successful, illustrated by the slightly

deformed shape, but the results are not as bad as H50L. As the specular hardness of the underlying object goes down (i.e., the specular lobe gets wider), we expect the specular filtering method to become less useful. This is due to the increased number of specular samples, implying that some specular samples will not be filtered out.

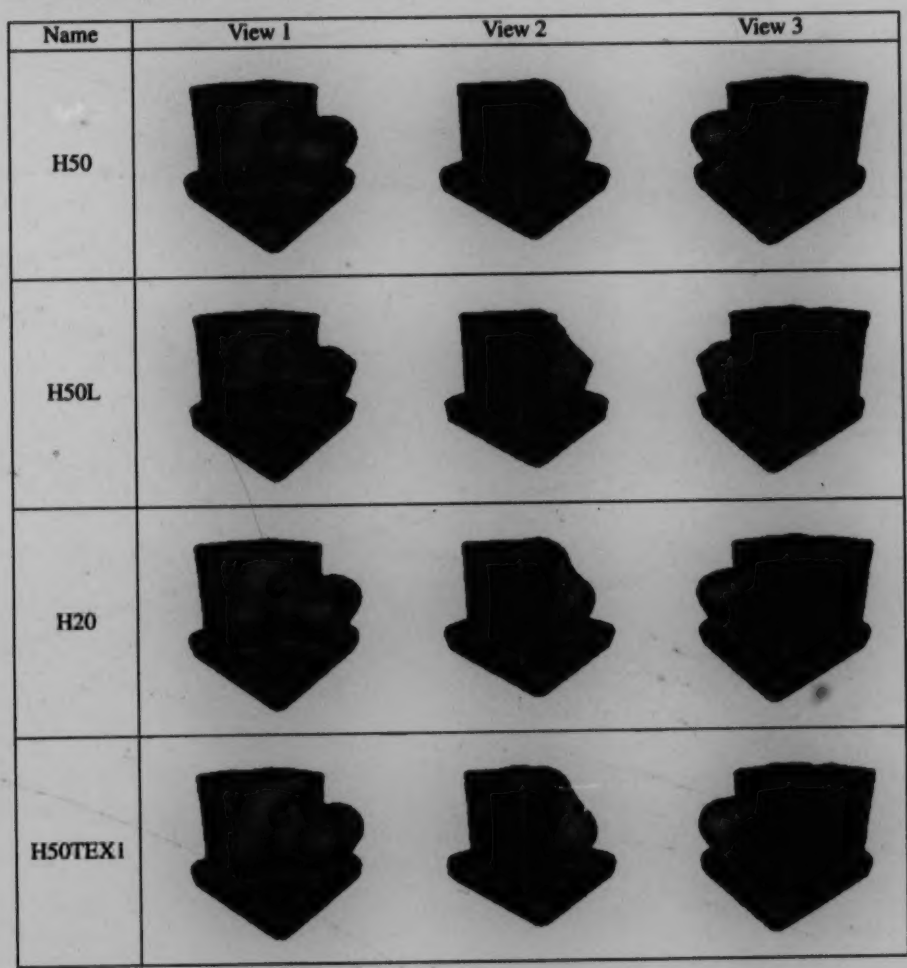

Figure 5.5: Results for the specular sequences. Comparing the results in H50 to those of H50L (no specular filtering), we can see that the specular filtering helps produce a better reconstruction. The shape recovery was also successful in the H20 sequence, and the H50TEX1 sequence.

5.2 Real Objects

In this section we will show the results of the shape optimization on real objects. We have chosen objects with varying surface properties (e.g., single matte material, textured diffuse object, and specular objects), to test the reconstruction method. In all cases the refinement stage takes roughly 20 minutes to 1 hour for completion[1]. The resulting mesh often has many little bumps, possibly due to image noise, so we often perform a simple mesh smoothing or reduce the resolution of the mesh after the optimization.

We first present the results for a single material, matte object: a 3D printout of the Stanford bunny. The sequence contains 52 images, and the shape optimization has been done with $n_{cameras} = 12$. Two input images, the ground truth, and the results are shown in Figure 5.6.

The refinement pulls the initial shape closer to the true object, making noticeable indentations by the legs of the bunny. Some of the small bumps also start to become visible through the refinement. Unfortunately, the small features, such as the ears and the base of the platform start to deteriorate slightly, although a more frequent application of the silhouette force could help overcome this problem.

In another experiment, we have captured two model houses, which both exhibit near Lambertian reflectance and have varying surface texture. The TallHouse capture consists of 52 images, 26 at each of the two light elevations, and the refinement was done with $n_{cameras} = 8$. The results are displayed in Figure 5.7. The refinement of the TallHouse brings out fine details, such as the stairs, the shingles on the roof, and the windows on the front.

The other model house, the ShortHouse, has a large concavity on the front of the object and another on the back (Fig. 5.8). The Gaussian pyramid implementation was required to reconstruct these concavities. In this sequence, we demonstrate typical problems that occur when fitting a spatially varying albedo to an erroneous shape. The second row of Figure 5.8 shows the fitted albedo using the initial geometry. The recovered texture is blurry in regions where the shape is not recovered, and does not produce a realistic rendering. In such cases, an image-based rendering technique could compensate for the geometric errors, but could only reproduce images from novel views that are close to input views. In this experiment, there are no images from above the object, so using an image-based method with the viewpoint in the first column would still produce erroneous renderings. Our results are shown in the bottom row of Fig 5.8, where we can see that the texture is much sharper than the results in the second row, and the overhead view is convincing.

5.2.1 Reflectance Results

We now present the results of objects having non-Lambertian appearance. Additionally, we show the results after fitting the Phong model to the surface.

[1] Experiments were run on $2.5 - 3.0$ GHz machines with ≥ 1 GB of RAM

Results of individual captures are presented in Figure 5.9. In each of the cases we can see that the reconstructed result closely resembles the input image. Comparing the novel renderings of the reconstructed Phong reflectance to the input images, it seems that the specular component is under fit. That is, the specular component is not as sharp as it is on the true object. A similar problem was noted in the work of Yu et al. [99]. As in their work, this could be a direct result of errors in the computed light position or errors in the recovered shape. Another problem is that our specular interpolation method often assigns a specular component to the regions of the object that are actually Lambertian.

Figure 5.10 demonstrates the practical usefulness of our models. In this scene, the elephant's legs and trunk have been animated so that it looks like the elephant is taking a drink from the coffee cup. Both the elephant and coffee cup have been reconstructed with the methods described in this thesis, and the two objects were easily combined with the synthetic sphere. The diffuse color of the objects has been reconstructed by the method in this paper, but the specular component was modified in Blender. The rendering, obtained by using the 3D modeler Blender, contains several light sources, realistic shadows, and demonstrates how these models can be modified and combined with traditional computer graphics models.

As another example of the potential applications of these models, we have captured a set of chess pieces. The unique chess pieces were individually captured and combined into a working chess game. Total time taken was 1.5 hrs for image acquisition and roughly 6 hours for refinement (a total of 12 models). Screenshots of the original chess board and the synthetic game are presented in Figure 5.11.

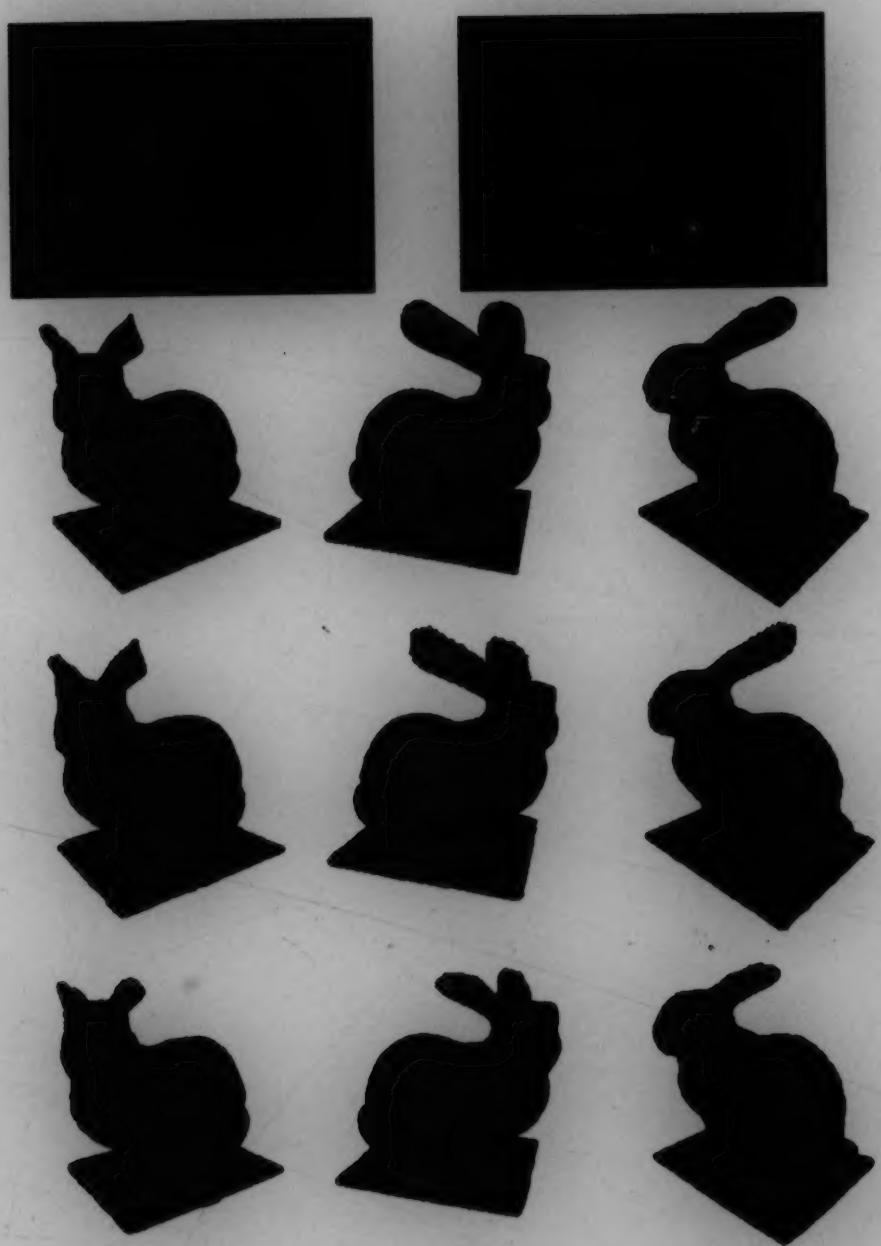

Figure 5.6: Two input images (top), synthetic images of the ground truth (2nd row), the initial volume (3rd row), and reconstruction results (bottom) from the Bunny sequence. The fine details start to become visible in the reconstructed results, mostly visible around the feet and between the legs of the bunny.

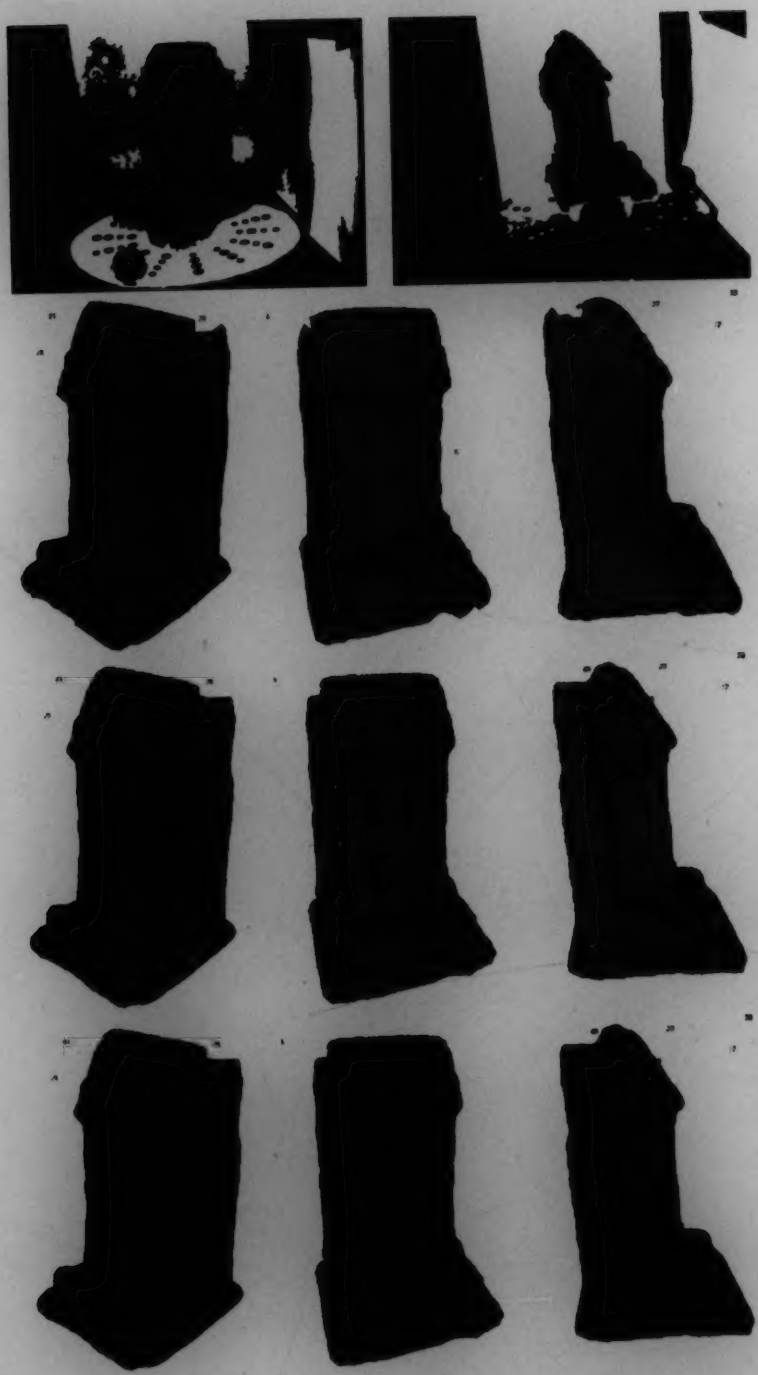

Figure 5.7: Two input images (top), the initial volume (2nd row), shaded reconstruction results (3rd row), and textured reconstruction results (bottom) from the TallHouse sequence. The refined results start to bring out the stairs in the base of the house, as well as the indentation by the chimney. The textured and lit renderings on the bottom row faithfully represent the true object.

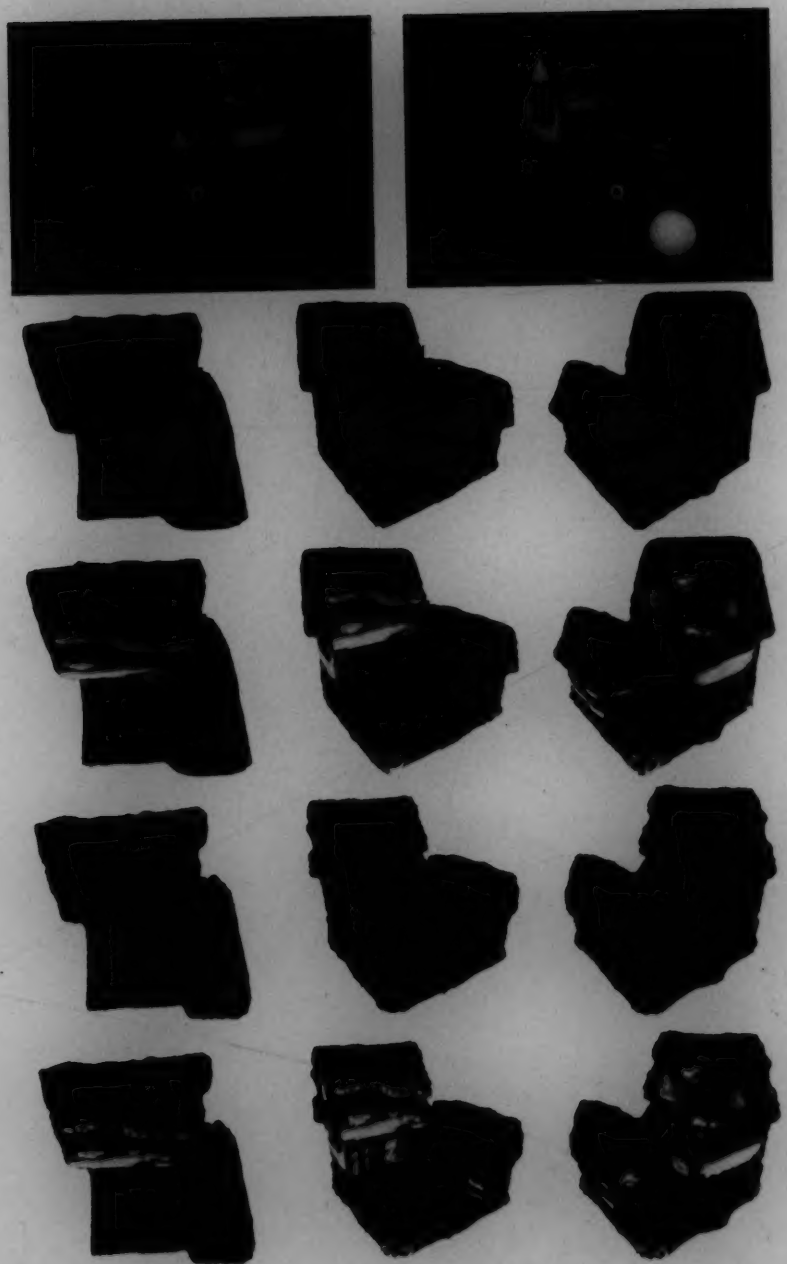

Figure 5.8: Two input images (top), the initial volume (2nd row), the albedo computed for the initial volume (3rd row), shaded reconstruction results (4th row), and albedo of reconstruction results (bottom) from the ShortHouse sequence. Notice the blurriness of the texture in the 2nd row resulting from inaccuracies in the shape. After the refinement, the recovered texture is sharp (4th row), and novel views that are far from input views are possible (1st column of 4th row).

Figure 5.9: Some results for shape estimation and reflectance fitting of specular objects. From left to right: an input image, a shaded image, the diffuse texture, diffuse and specular texture at the same viewpoint, and a novel viewpoint with shadows and two lights. The renderings of each object are realistic, although the specular component is not as sharp as observed in the input images.

Figure 5.10: Multiple captures (real images of the two objects are given on the top row) were combined with some synthetic objects (the sphere and ground plane) and then rendered in Blender with multiple novel lights and a novel viewpoint (bottom). The vertices of the elephant have been modified to demonstrate the practical usefulness of the models.

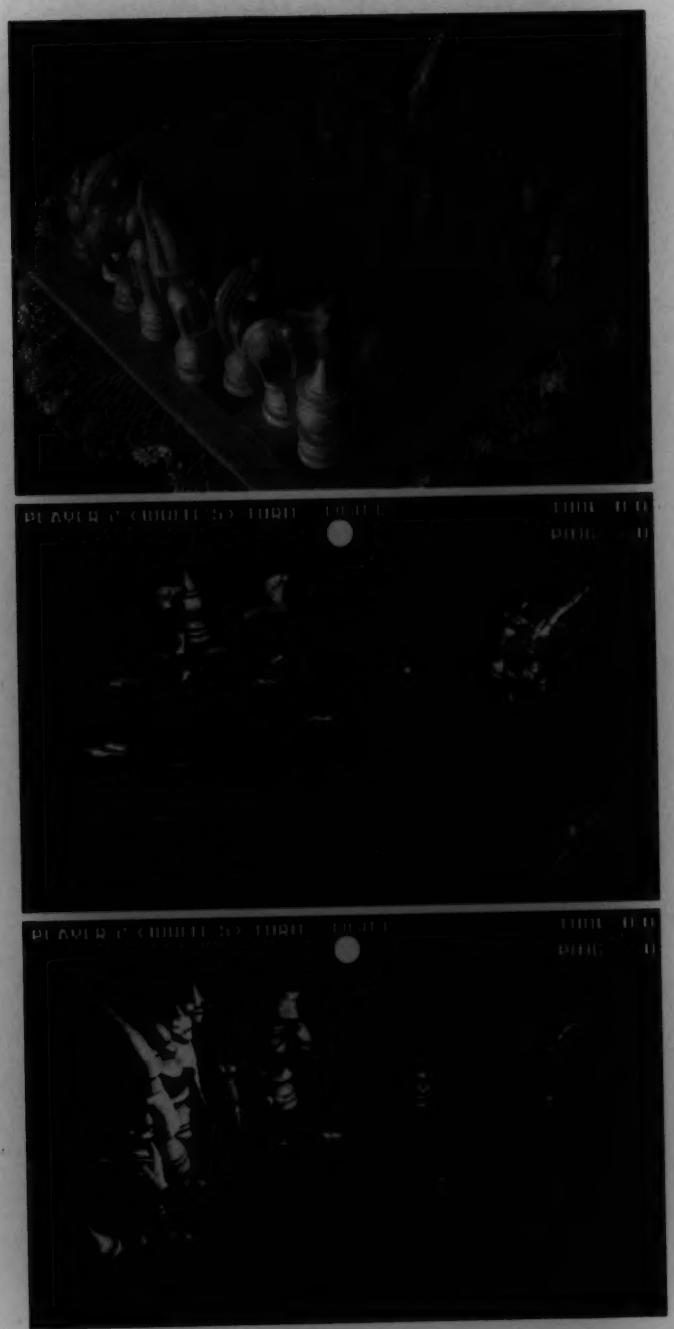

Figure 5.11: The unique pieces of the real chess board (top) were captured individually. These pieces were then combined to produce a working chess game. The middle and bottom images show two lighting conditions of the chess game, where the objects are both casting and receiving shadows.

Chapter 6

Conclusions

In this thesis we have presented a capture setup for obtaining the shape and reflectance of solid 3D models from images. The system has been designed to be easy to use, while operating with minimal hardware requirements. In the simplest case, we have captured objects using a rotating cake table, a single consumer handheld camera, a desktop lamp, a painted ping-pong ball, and a couple sheets of blue paper. Each module of the system has been presented with enough detail to allow for an independent implementation.

Within this capture setup, one of the main contributions was the use of a consistency function that incorporates a known light position. Assuming that the object was moving relative to the light allowed the recovery of spatially varying reflectance parameters. The consistency measure fits nicely into the variational framework, which leads to a PDE describing the evolution of a surface. In our implementation, we have represented the surface using a triangulated mesh, which is deformed from an initial volume, the visual hull, to the final refined shape. Instead of a mesh, we would like to consider the use of a level set representation during the reconstruction. Another implementation option that we have not yet explored is to measure the photo-consistency on the tangent plane of the vertices (mentioned in Chapter 3), instead of on the triangles themselves.

We have demonstrated that our implementation is effective for textureless and low textured objects, but seems to be less useful on high frequency texture. Unfortunately, the method also seems limited in the range of concavities that can be recovered. In order to recover large concavities, the refinement needs to be started with a low resolution mesh and downsampled images. In such circumstances the use of a silhouette preserving force was necessary to preserve the volume of the object during the reconstruction (similar observations were made by Isidoro and Sclaroff [37]). Fortunately, these textured regions are where correlation based methods are expected to work well [23]. A possible alternative to the use of a low resolution mesh would be to directly carve high textured regions (as done in [22]), while allowing the gradient based refinement to do the remaining portions of the object.

As mentioned above, the silhouette force was necessary when using a low resolution mesh. In the current implementation the silhouette force is a simple ad-hoc approach that does not fit nicely

into the variational framework used. We would like to investigate a mathematical measurement of the photo-consistency in the images, where it is more natural to represent a silhouette preserving force. In this formulation, we suspect that the silhouette preserving force could be encoded directly in the PDE evolution. The work of Yezzi and Soatto on *stereoscopic segmentation* [98] is a potential starting point for this extension.

A downside of the current implementation is that the quality of the final results is directly dependent on the initial resolution of the mesh, and the other parameters, such as $n_{cameras}$. As the refinement typically takes anywhere from 20 minutes to 1 hour, it takes time to adjust these parameters. We would like to further investigate the effects of the parameters and the initial mesh resolution to reduce the number of runs required to get good final results.

Specular highlights have also been accounted for by treating them as outliers, allowing the method to recover the shape of glossy objects. We have demonstrated that the shape is successfully recovered for some specular objects, but we suspect the reliability of our approach to degrade with materials having a wide specular lobe. A potential direction for future work is to utilize the information contained in the specular highlights, rather than filtering them out.

An additional area for future efforts is the integration of a user into the shape recovery. In some reconstructions, it is clear to a user that a particular portion of the object should be planar, or that an edge should be 90 degrees. We would like to introduce these constraints to allow optional user intervention during the reconstruction process. The model would then be refined to best match the input images, while maintaining the user defined constraints.

Unfortunately, the capture system described in this work is limited to indoor settings and small capture objects. To make the methods more practical, we would like to investigate similar capture schemes for large scale 3D models in outdoor environments. In outdoor environments, introducing light variation is not a feasible option, so instead we would like to consider dense shape and reflectance reconstruction where the light variation comes from captures at different times of day.

Appendix A

Sampling the occluding contour of a sphere

The projection of a sphere onto an image plane is generally an ellipse. One approach for determining sample points for the occluding contour would be to analytically derive the equation of this ellipse, and choose sample points directly on this ellipse. Alternatively, one could determine points on the occluding contour in 3D and project these points onto the image. We take the latter approach.

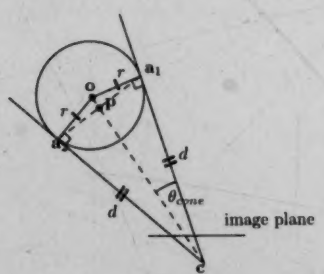

Figure A.1: If $\mathbf{a}_1$ and $\mathbf{a}_2$ are on the occluding contour of the sphere then $\overline{\mathbf{a_2o}} = \overline{\mathbf{a_1o}} = r$, $\angle\mathbf{oa_2c} = \angle\mathbf{oa_1c}$, and $|\mathbf{co}| = |\mathbf{co}| \Rightarrow \triangle\mathbf{oa_2c} = \triangle\mathbf{oa_1c} \Rightarrow \overline{\mathbf{a_2c}} = \overline{\mathbf{a_1c}}$.

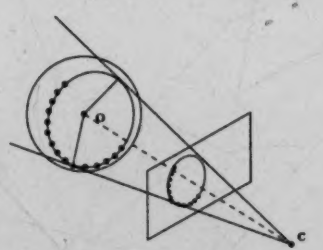

Figure A.2: The occluding contour of a sphere is a circle on the sphere, which is projected into the image.

If $\mathbf{x}$ is a point on a sphere of radius r, $\mathbf{c}$ is the center of the camera, and the ray $\vec{r}(t) = \mathbf{c}+t(\mathbf{x}-\mathbf{c})$ is tangent to the sphere, then $\mathbf{x}$ is on the occluding contour of the sphere. It is not hard to show that all such points actually lie on the intersection of a plane with the sphere. This is illustrated in Fig. A.1, where by similar triangles, all points on the occluding contour of the sphere are equidistant from the camera center c.

The plane is defined by the normal, which is aligned with $\mathbf{o} - \mathbf{c}$, and the point $\mathbf{p}$ on the line joining $\mathbf{c}$ to $\mathbf{o}$. By Pythagoras theorem, all points on the occluding contour are distance $d = \sqrt{\|\mathbf{o} - \mathbf{c}\|^2 - r^2}$ from the camera center. This distance defines the angle, $\theta_{cone} = \tan^{-1}(\frac{r}{d})$, of the cone from $\mathbf{c}$ through the occluding contour. The point $\mathbf{p}$ is then given by the orthogonal projection of an occluding contour point onto the line $\overline{\mathbf{oc}}$, i.e., $\|\mathbf{p} - \mathbf{c}\|=d\cos(\theta_{cone}) \Rightarrow \mathbf{p} =$

$\mathbf{c} + d\cos(\theta_{cone})\frac{(\mathbf{o}-\mathbf{c})}{\|\mathbf{o}-\mathbf{c}\|}$.

Once the plane is found, we sample points at regular angular intervals on the intersection of this plane with the sphere, which is a circle on the plane. The radius is readily obtained as $r_{circ} = \sqrt{d^2 - \|\mathbf{p}-\mathbf{c}\|^2}$. These points are then projected onto the corresponding image (Fig. A.2).

Bibliography

[1] Tomas Akenine-Möller and Eric Haines. *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 2nd edition, 2002.

[2] Javier Iglesia Aparicio and Jaime Gómez García-Bermejo. An approach for determining phong reflectance parameters from real objects. In *ICPR*, pages 3572–3575, 2000.

[3] Adam Baumberg, Alex Lyons, and Richard Taylor. 3D S.O.M. - a commercial software solution to 3d scanning. In *Proceedings of Vision, Video, and Graphics (VVG'03)*, pages 41–48, July 2003.

[4] Bruce G. Baumgart. A polyhedron representation for computer vision. In *AFIPS National Conference*, volume 44, pages 589–596, 1975.

[5] Peter N. Belhumeur, David J. Kriegman, and Alan L. Yuille. The bas-relief ambiguity. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1060, Washington, DC, USA, 1997. IEEE Computer Society.

[6] D.N. Bhat and S.K. Nayar. Stereo in the presence of specular reflection. In *Proceedings of International Conference on Computer Vision (ICCV'95)*, pages 1086–1092, 1995.

[7] James F. Blinn. Models of light reflection for computer synthesized pictures. In *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 192–198, New York, NY, USA, 1977. ACM Press.

[8] Thomas Bonfort and Peter Sturm. Voxel carving for specular surfaces. In *Proceedings of the 9th IEEE International Conference on Computer Vision*. IEEE CSP, october 2003.

[9] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on PAMI*, 23(11):1222–1239, 2001.

[10] Michael J. Brooks and Berthold K.P. Horn. Shape and source from shading. In *MIT AI Memo*, 1985.

[11] Vincent Caselles, Ron Kimmel, Guillermo Sapiro, and Catalina Sbert. Minimal surfaces based object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):394–398, 1997.

[12] Young-Chang Chang and J.F. Reid. Rgb calibration for color image analysis in machine vision. *IEEE Transactions on Image Processing*, 5(10):1414–1422, October 1996.

[13] Vikram Chhabra. Reconstructing specular objects with image based rendering using color caching. Master's thesis, WORCESTER POLYTECHNIC INSTITUTE, may 2001.

[14] Dana Cobzas, Keith Yerex, and Martin Jägersand. Dynamic textures for image-based rendering of fine-scale 3d structure and animation of non-rigid motion. *Computer Graphics Forum*, 21(3), 2002.

[15] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Trans. Graph.*, 23(3):905–914, 2004.

[16] J. Cryer, P. Tsai, and M. Shah. Combining shape from shading and stereo using human vision model. Technical Report CSTR-92-25, University of Central Florida, 1992.

[17] W. Bruce Culbertson, Thomas Malzbender, and Gregory G. Slabaugh. Generalized voxel coloring. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 100–115, London, UK, 2000. Springer-Verlag.

[18] Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[19] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[20] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Proceedings of SIGGRAPH 96*, pages 11–20, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[21] Ye Duan, Liu Yang, Hong Qin, and Dimitris Samaras. Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. In *ECCV (3)*, pages 238–251, 2004.

[22] Carlos Hernández Esteban and F. Schmitt. Multi-stereo 3d object reconstruction. In *1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'02)*, pages 159–166, June 2002.

[23] Carlos Hernández Esteban and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. In *4th International Conference on 3D Digital Imaging and Modeling (3DIM'03)*, pages 46–53, October 2003.

[24] Carlos Hernández Esteban and F. Schmitt. A snake approach for high quality image-based 3d object modeling. In *2nd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, pages 241–248, October 2003.

[25] Olivier Faugeras and Renaud Keriven. Variatonal principles, surface evolution, pdes, level set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7:336–344, Mar 1998.

[26] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[27] Robert T. Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(4):439–451, 1988.

[28] P. Fua and Y. G. Leclerc. Object-centered surface reconstruction: combining multi-image stereo and shading. *Int. J. Comput. Vision*, 16(1):35–55, 1995.

[29] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[30] Jaime Gómez García-Bermejo and J. López Coronado. F. J. Díaz Pernas. An approach for determining bidirectional reflectance parameters from range and brightness data. In *IEEE International Conference on Image Processing (ICIP'96)*, pages 41–44, 1996.

[31] Athinodoros S. Georghiades. Incorporating the torrance and sparrow model of reflectance in uncalibrated photometric stereo. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2, pages 816–823, October 2003.

[32] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1996. ACM Press.

[33] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[34] H. Hayakawa. Photometric stereo under a light-source with arbitrary motion. *JOSA-A*, 11(11):3079–3089, November 1994.

[35] Aaron Hertzmann and Steven M. Seitz. Shape and materials by example: A photometric stereo approach. In *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR'03)*, volume 1, pages 533–540, June 2003.

[36] Katsushi Ikeuchi and Kosuke Sato. Determining reflectance properties of an object using range and brightness images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1139–1153, 1991.

[37] J. Isidoro and S. Sclaroff. Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints. In *Proceedings of International Conference on Computer Vision (ICCV'03)*, pages 1335–1342, 2003.

[38] Hailin Jin, Daniel Cremers, Anthony J. Yezzi, and Stefano Soatto. Shedding light on stereoscopic segmentation. In *Computer Vision and Pattern Recognition (CVPR 2004)*, volume 1, pages 36–42, 2004.

[39] Hailin Jin, Stefano Soatto, and Anthony J. Yezzi. Stereoscopic shading: Integrating multiframe shape cues in a variational framework. In *Computer Vision and Pattern Recognition (CVPR 2000)*, volume 1, pages 1169–1176, 2000.

[40] Hailin Jin, Stefano Soatto, and Anthony J. Yezzi. Multi-view stereo beyond lambert. In *In Proc. IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pages 171–178, June 2003.

[41] Hailin Jin, Anthony J. Yezzi, and Stefano Soatto. Variational multiframe stereo in the presence of specular reflections. In *3DPVT*, pages 626–631, 2002.

[42] David Knossow, Remi Ronfard, Frédéric Devernay, and Radu Horaud. Tracking with inverse kinematics of apparent contours. Submitted to ICCV 2005.

[43] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision (ECCV'02)*, volume 3, pages 82–96, May 2002.

[44] K. N. Kutulakos and Steven Seitz. A theory of shape by space carving. Technical Report TR692, Computer Science Deptartment, University of Rochester, May 1998.

[45] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *Int. J. Comput. Vision*, 38(3):199–218, 2000.

[46] J.-O. Lachaud and A. Montanvert. Deformable meshes with automated topology changes for coarse-to-fine 3D surface extraction. *Medical Image Analysis*, 3(2):187–207, 1999.

[47] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 117–126, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[48] Holger Lange. Advances in the cooperation of shape from shading and stereo vision. In *3DIM*, pages 46–58, 1999.

[49] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(2):150–162, 1994.

[50] John M. Lee. *Riemannian Manifolds: An Introduction to Curvature*. Springer, 1997.

[51] Hendrik P. A. Lensch, Michael Goesele, Jan Kautz, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatially varying materials. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 103–114, London, UK, 2001. Springer-Verlag.

[52] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatial appearance and geometric detail. *ACM Trans. Graph.*, 22(2):234–257, 2003.

[53] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, New York, NY, USA, 1996. ACM Press.

[54] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 362–371, New York, NY, USA, 2002. ACM Press.

[55] Robert R. Lewis. Making shaders more physically plausible. *Computer Graphics Forum*, 13(2):109–120, 1994.

[56] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM Press.

[57] Sebastian Magda, David J. Kriegman, Todd Zickler, and Peter N. Belhumeur. Beyond lambert: Reconstructing surfaces with arbitrary brdfs. In *Proceedings 8th IEEE International Conference on Computer Vision*, volume 2, pages 391–398, June 2001.

[58] Stephen Robert Marschner. *Inverse rendering for computer graphics*. PhD thesis, Cornell University, 1998. Adviser-Donald P. Greenberg.

[59] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, 2002.

[60] Y. Matsumoto, K. Fujimura, and T. Kitamura. Shape-from-silhouette/stereo and its application to 3-d digitizer. In *DCGI '99: Proceedings of the 8th International Conference on Discrete Geometry for Computer Imagery*, pages 177–190, London, UK, 1999. Springer-Verlag.

[61] Wojciech Matusik, Chris Buehler, and Leonard McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 115–126, London, UK, 2001. Springer-Verlag.

[62] Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. Efficient isotropic brdf measurement. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 241–247. Eurographics Association, 2003.

[63] T. Mitsunaga and S.K. Nayar. Radiometric self calibration. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 374–380, 1999.

[64] S. K. Nayar. Sphereo: Determining depth using two specular spheres and a single camera. In *Proceedings of SPIE Conference on Optics, Illumination, and Image Sensing for Machine Vision III*, pages 245–254, November 1988.

[65] K. Nishino, Yoichi Sato, and Katsushi Ikeuchi. Eigen-texture method: appearance compression based on 3d model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, volume 1, pages 618 – 624, June 1999.

[66] K. Nishino, Z. Zhang, and K. Ikeuchi. Determining reflectance parameters and illumination distribution from a sparse set of images for view-dependent image synthesis. In *Proceedings of Eighth IEEE International Conference on Computer Vision (ICCV '01)*, volume 1, pages 599–606, July 2001.

[67] David Nistér. Automatic passive recovery of 3d from images and video. In *Second International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT04)*, pages 438–445, September 2004.

[68] S. Nobuhara and T. Matsuyama. Dynamic 3d shape from multi-viewpoint images using deformable mesh models. In *Proceedings of 3rd International Symposium on Image and Signal Processing and Analysis*, pages 192–197, September 2003.

[69] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[70] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[71] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[72] Mark Pollefeys, R. Koch, and L. Van Gool. A simple and efficient rectification method for general motion. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, pages 496–501, 1999.

[73] Mark W. Powell, Sudeep Sarkar, and Dmitry Goldgof. A simple strategy for calibrating the geometry of light sources. *IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions*, 23(9):1022–1027, September 2001.

[74] Hossein Ragheb and Edwin R. Hancock. Separating lambertian and specular reflectance components using iterated conditional modes. In *BMVC*, 2001.

[75] C. Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno. Marching intersections: An efficient resampling algorithm for surface management. In *SMI '01: Proceedings of the International Conference on Shape Modeling & Applications*, page 296, Washington, DC, USA, 2001. IEEE Computer Society.

[76] Hideo Saito, Kazuko Omata, and Shinji Ozawa. Recovery of shape and surface reflectance of specular object from rotation of light source. In *2nd International Conference on 3D Digital Imaging and Modeling (3DIM '99)*, pages 526–535. IEEE Computer Society, 1999.

[77] Yoichi Sato, Mark D. Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 379–387, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[78] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, 2002.

[79] Hartmut Schirmacher, Wolfgang Heidrich, Martin Rubick, Detlef Schiron, and Hans-Peter Seidel. Image-based brdf reconstruction. In *Proceedings of the 4th Conference on Vision, Modeling, and Visualization (VMV-99)*, pages 285–292, November 1999.

[80] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1067, Washington, DC, USA, 1997. IEEE Computer Society.

[81] Khurram Shafique and Mubarak Shah. Estimation of the radiometric response functions of a color camera from differently illuminated images. In *IEEE International Conference on Image Processing*, October 2004.

[82] Denis Simakov, Darya Frolova, and Ronen Basri. Dense shape reconstruction of a moving object under arbitrary, unknown lighting. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1202, Washington, DC, USA, 2003. IEEE Computer Society.

[83] Gregory G. Slabaugh, Ronald W. Schafer, and Mat C. Hans. Multi-resolution space carving using level sets methods. In *IEEE International Conference on Image Processing*, pages 545–548, 2002.

[84] Stefano Soatto, Anthony J. Yezzi, and Hailin Jin. Tales of shape and radiance in multi-view stereo. In *ICCV03*, pages 974–981, 2003.

[85] Tatiana Surazhsky, Evgeny Magid, Octavian Soldea, Gershon Elber, and Ehud Rivlin. A comparison of gaussian and mean curvatures triangular meshes. In *Proceedings of IEEE International Automation (ICRA2003)*, pages 1021–1026, Taipei, Taiwan, 14-19 September 2003.

[86] Richard Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Underst.*, 58(1):23–32, 1993.

[87] Marco Tarini, Marco Callieri, Claudio Montani, Claudio Rocchini, Karin Olsson, and Therese Persson. "marching intersections: An efficient approach to shape-from-silhouette". In Bernd Girod, Günther Greiner, Heinrich Niemann, and Hans-Peter Seidel, editors, *Proceedings of the Conference on Vision, Modeling, and Visualization (VMV 2002)*, pages 255–262, November 2002.

[88] Kenneth E. Torrance and E.M. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America*, 57(9):1105–1114, September 1967.

[89] Adrien Treuille, Aaron Hertzmann, and Steven M. Seitz. Example-based stereo with general brdfs. In Tomás Pajdla and Jiri Matas, editors, *ECCV (2)*, volume 3022 of *Lecture Notes in Computer Science*, pages 457–469. Springer, 2004.

[90] George Vogiatzis, Philip Torr, and Roberto Cipolla. Bayesian stochastic mesh optimisation for 3d reconstruction. In *Proceedings of British Machine Vision Conference (BMVC'03)*, pages 711–718, 2003.

[91] George Vogiatzis, Philip Torr, S.M. Seitz, and Roberto Cipolla. Reconstructing relief surfaces. In *Proceedings of British Machine Vision Conference (BMVC'04)*, pages 117–126, 2004.

[92] Martin Weber, Andrew Blake, and Roberto Cipolla. Towards a complete dense geometric and photometric reconstruction under varying pose and illumination. In *BMVC*, 2002.

[93] Lawrence B. Wolff and Elli Angelopoulou. Three dimensional stereo by photometric ratios. *Journal of the Optical Society of America A: Optics, Image Science, and Vision (JOSA)*, 11(11):3069–3078, November 1994.

[94] Robert J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[95] Robert J. Woodham. Gradient and curvature from the photometric stereo method, including local confidence estimation. *Journal of the Optical Society of America A*, 11(11):3050–3068, 1994.

[96] Enhua Wu, Qimin Sun, and Xuehui Liu. Recovery of material under complex illumination conditions. In *GRAPHITE '04: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and Southe East Asia*, pages 39–45. ACM Press, 2004.

[97] Ruigang Yang, Marc Pollefeys, and Greg Welch. Dealing with textureless regions and specular highlights-a progressive space carving scheme using a novel photo-consistency measure. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 576, Washington, DC, USA, 2003. IEEE Computer Society.

[98] Anthony J. Yezzi and Stefano Soatto. Stereoscopic segmentation. In *International Conference on Computer Vision (ICCV 2001)*, pages 59–66, 2001.

[99] Tianli Yu, Ning Xu, and Narendra Ahuja. Recovering shape and reflectance model of non-lambertian objects from multiple views,. In *Proceedings of Computer Vision and Pattern Recognition (CVPR'04)*, volume 2, pages 226–233, February 2004.

[100] Tianli Yu, Ning Xu, and Narendra Ahuja. Shape and view independent reflectance map from multiple views. In *ECCV (4)*, pages 602–616, 2004.

[101] X. Zabulis and K. Daniilidis. Multi-camera reconstruction based on surface normal estimation and best viewpoint selection. In *Proceedings of 2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'04)*, pages 733–740, September 2004.

[102] Christopher Zach, Andreas Klaus, Markus Hadwiger, and Konrad Karner. Accurate dense stereo reconstruction using graphics hardware. In *Eurographics 2003 Short Presentations*, pages 227–234, 2003.

[103] Huaifeng Zhang, Jan Cech, Radim Sara, Fuchao Wu, and Zhanyi Hu. A linear trinocular rectification method for accurate stereoscopic matching. In *Proceedings of British Machine Vision Conference(BMVC'03)*, pages 281–290, 2003.

[104] Li Zhang, Brian Curless, Aaron Hertzmann, and Steven M. Seitz. Shape and motion under varying illumination: Unifying structure from motion, photometric stereo, and multi-view stereo. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV'03)*, Oct 2003.

[105] Li Zhang and Steven M. Seitz. Image-based multiresolution shape recovery by surface deformation. In *Proceedings of SPIE: Videometrics and Optical Methods for 3D Shape Measurement*, January 2001.

[106] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(8):690–706, 1999.

[107] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision (ICCV'99)*, pages 666–673, September 1999.

[108] Todd Zickler, Peter N. Belhumeur, and David J. Kriegman. Helmholtz stereopsis: Exploiting reciprocity for surface reconstruction. In *Proceedings 7th European Conference on Computer Vision*, volume 3, pages 869–884, May 2002.